TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky a mezioborových inženýrských studií

Studijní program: M 2612 – Elektrotechnika a informatika
Studijní obor: 3902T005 – Automatické řízení a inženýrská informatika

Detekce nečistot stravitelných střev

Detection of Stains on Sausage Casings

Diplomová práce

Autor: **František Kratochvíl**
Vedoucí DP: Doc. Ing. Ivan Jaksch, CSc.
Konzultant: Ing. Stanislav Mrázek, Cutisin a.s.
Ing. Lukáš Matela

V Liberci, 1.5.2006

Tato stránka je prázdná a na její místo bude vloženo zadání

Prohlášení

Byl jsem seznámen s tím, že na mou diplomovou práci se plně vztahuje zákon č. 12/2000 o právu autorském, zejména § 60 (školní dílo).

Beru na vědomí, že TUL má právo na uzavření licenční smlouvy o užití mé DP a prohlašuji, že **s o u h l a s í m** s případným užitím mé diplomové práce (prodej, zapůjčení apod.).

Jsem si vědom toho, že užít své diplomové práce či poskytnout licenci k jejímu využití mohu jen se souhlasem TUL, která má právo ode mne požadovat přiměřený příspěvek na úhradu nákladů vynaložených univerzitou na vytvoření díla (až do jejich skutečné výše).

Diplomovou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím diplomové práce a konzultantem.

Datum: 1.5.2006

Podpis:

Poděkování

Práce byla napsána s použitím české verze systému LaTeX, jeho grafického prostředí pro Unix/Linux Kile, grafického programu Gimp a dalších volně dostupných programů pro operační systém GNU/Linux. Chtěl bych tímto poděkovat jejich autorům.

Děkuji konzultantu Ing. Lukaši Matelovi za jeho rady a kontrolu diplomové práce a Slávce Franclíkové za pomoc s typografickými pravidly.

Abstrakt

Diplomová práce se zabývá problematikou detekce vad na výstupu procesu výroby umělých střev. Kontrola kvality výrobků je důležitou součástí každého výrobního procesu, obzvláště v potravinářském průmyslu. Nahradit člověka v roli kontrolora automatikou je však v mnoha případech stále obtížné, zejména jedná-li se o veličiny, které se nedají žádným obvyklým způsobem měřit.

Při výrobě umělých střev určených pro výrobu uzenin může dojít z různých důvodů ke znečištění výsledného produktu. Zdroji znečistění bývají různé látky a materiály, které jsou používány při samotné výrobě nebo při údržbě strojů; další nečistoty se do výroby mohou dostat selháním lidského faktoru. Tyto vady se na výrobcích projevují tmavými skvrnami náhodného rozmístění a tvaru a jsou proto opticky zjistitelné. Jejich nalezení metodami analýzy obrazu komplikuje struktura povrchu střeva, která se vytvoří při řásnění. Tato struktura se na obraze projevuje vzestupy a poklesy jasu, které mají často jasovou úroveň shodnou s úrovní jasu vad.

V rámci diplomové práce je prozkoumáno několik způsobů detekce vad, které používají různé metody analýzy obrazových signálů. Kromě spolehlivosti metody v odhalení vady je kladen důraz také na robustnost řešení a nízkou výpočetní náročnost při provádění programu. Každá z použitých metod je vyzkoušena na několika vzorcích, které reprezentují výrobky s různými druhy a velikostmi vad; metoda s nejlepšími výsledky je podrobena testu na větším množství vzorků.

Na počátku detekce nečistot je pořízení snímků kamerou. Následuje segmentace objektu od pozadí, která izoluje oblast zájmu pro další zpracování. Další postup se liší použitou metodou. Prvním algoritmem je analýza textur pomocí Kohonenovy nervové sítě, která je vytvořena na základě vybraného souboru vzorků a následně použita ke klasifikaci textur. Další metodou je hledání lokálních extrémů jasové funkce s využitím jednoduchého algoritmu detekce hran a obrazových filtrů. Třetí metoda využívá nelineárních obrazových filtrů šedotónové matematické morfologie.

Právě poslední zmíněný algoritmus slibuje dobré výsledky detekce nečistot. V laboratorních testech dokázal správně určit vysoké procento testovaných vzorků různých rozměrů a s různými vadami. Další postup by se proto měl soustředit na tuto metodu.

Klíčová slova: zpracování obrazu, šedotónová morfologie, Kohonenovy mapy

Abstract

The goal of this thesis is to offer a way of detecting flaws of sausage casings at the end of the manufacturing process. Quality control is an important part of every industrial process, especially in food industry. It becomes difficult to replace human in the role of supervisor with automation in cases where the important properties of the product are not measurable in any usual way.

During sausage casings manufacturing process, the product may become flawed from various reasons. The sources of admixtures are often substances and materials used in the manufacturing process or in maintenance. Other flaws may appear as a consequence of human failure. The flaws can be recognized by dark spots on the casings of various shapes and tones and can therefore be optically distinguished. The structure of the casings which is introduced in the compacting process makes surface flaw detection based on image analysis more complicated.

Several ways of flaw detection are suggested and examined in the thesis, all of them depending on image analysis. Together with reliability of the solution, the focus is made on making the method robust and not computationally complex. Each of the methods is tried on a small set of samples, which represent products with various sized and shaped flaws. A method which shows the best results is tested on a wider set of samples.

Flaw detection process starts with aquiring images using a camera. Next, objects are segmented from the background and the region of interest for next analysis is set. The following process differs from the used method. First algorithm uses Kohonen's Self-Organizing Maps, the maps being created on a chosen set of samples and then used for texture classification. Second method searches for local extremes in brightness function using a simple edge-detection algorithm. Third method uses nonlinear image filters of graylevel mathematical morphology.

The third mentioned algorithm promises the best results in detecting flaws. During laboratory tests, a high percent of tested samples of various sizes and with various flaws were succesfully figured out. Additional work should therefore focus on this method.

Keywords: image analysis, greylevel morphology, self-organizing maps

Obsah

Úvod

Firma Cutisin, s.r.o vyrábí bílkovinné obaly pro masný a uzenářský průmysl na bázi přírodního materiálu. Základní surovinou je vnitřní vrstva hovězí kůže – vláknitá bílkovina složená převážně z kolagenu.

Při výrobě se na kolagenních střevech z různých příčin mohou objevit nečistoty. Jedná se o skvrny od oleje a mazadel z výrobních zařízení, skvrny způsobené znečištěním základního výrobního materiálu a další nečistoty, které se mohou dostat do výroby např. nepozorností pracovníka. Ačkoliv dochází k výskytu takových nečistot velmi zřídka, snahou výrobce je tyto případy úplně eliminovat a zamezit tak situaci, kdy by se znečištěný produkt mohl dostat do další výroby.

Cílem této diplomové práce je navrhnout detekci takovýchto vad na základě analýzy obrazu.

1 Výroba bílkovinných obalů

1.1 Zpracování základní suroviny

Základní surovinou je klihovka – část kůže skotu. Chemickým zpracováním klihovky se získá kolagenní hmota pro výrobu umělých střev.

Extruze kolagenních střev

Připravená kolagenní hmota je přiváděna do extruzní hlavy tažného stroje, kde se vytváří nekonečná bezešvá hadice – střevo. To je pak vedeno dále do sušící linky, kde probíhá jeho termické vysušení.

Tažení střeva

V tažném stroji se vysokým tlakem protlačuje kolagenní hmota štěrbinou extruzní hlavy, kde je hmota vytvarována do vzduchem naplněné trubice o stanoveném průměru. Při tom jsou v rotační části hlavy kolagenní vlákna uspořádána do mřížky. Extruzní hlava je určena vždy pro výrobu určitého průměru (kalibru) střeva. Zásobník hmoty, pracovní válce a extruzní hlava jsou chlazeny studenou vodou, aby nedošlo k degradaci kolagenu.

Sušení na sušící lince

Vytvarované střevo přechází z extruzní hlavy do sušící linky, kde je unášeno soustavou hnaných nosných válečků a kol. Teplota vzduchu přiváděného do sušícího kanálu je měřena tzv. suchým teploměrem (udává teplotu vzduchu) a mokrým teploměrem (udává teplotu stěny střeva). Po usušení střeva na určitou vlhkost je střevo převedeno na sušící dráhu, kde probíhá jeho intenzivní dosušení a postřik. Ze suchých drah je střevo vedeno přes laserový měřič průměru do kapsy s přítlačnými válečky, kde se vytlačuje vzduch. Složené střevo je navíjeno na roli.

Střevo je dle typu postřikováno pitnou vodou nebo neutralizační kapalinou společně s tvrdící kapalinou. Neutralizační postřik ovlivňuje pH hotového výrobku. Tvrdící postřik ovlivňuje jeho pevnost a také jej vyhlazuje.

Úprava salámových střev

Převíjení – vyrobené střevo se na tažebně navíjí na velké kovové role. Na těchto rolích je dopraveno do skladu zrání na úpravně. Poté je střevo převíjeno na papírové dutinky. Před expedicí se roličky balí do polyetylenových sáčků.

Úprava jedlých střev řásněním

Jedlá střeva navinutá na širokých rolích jsou na tažebně navlhčena na vlhkost optimální pro řásnění. Střevo je odvíjeno z rolí a vedeno do podtlakové komory, kde je navléknuto na tyč, olejováno a vedeno podávacími kladkami do řásnící hlavy. Z hlavy přechází nařásněné střevo na řásnící trn, který určuje světlost roubíku. Po odstřižení je tzv. roubík posunut na otočný revolver, kde je lisován. Roubíky jsou po řásnění ukládány do krabic.

2 Úvodní rozbor problému

Jedná se o problém kontroly výstupní kvality při výrobě umělých párkových střev. Produktem výroby je *roubík* – umělé střevo určené pro výrobu uzenin, které je upravené řásněním do kompaktní podoby určených rozměrů. Opticky viditelné vady v podobě tmavých skvrn se vyskytují již na vstupu do řásnícího stroje. Po zřásnění je určité procento vad skryto uvnitř záhybů roubíku.

Máme tedy tyto možnosti umístění bodu kontroly kvality:

- Střevo v podobě hadice po extruzi
- Střevo v plošné podobě před řásněním
- Jednotlivé roubíky na výstupu z řásnícího stroje

2.1 Možnosti umístění kontroly kvality po extruzi

Po extruzi a sušení má střevo podobu hadice. Pohybuje se rychlostí v řádu metrů za sekundu – pro zachycení detailů povrchu je tedy nutno použít řádkové kamery. Má-li střevo kruhový průřez, jsou potřeba nejméně tři kamery, které rozmístíme rovnoměrně kolem střeva. V takto pořízeném obraze lze pak snadno ve zlomku sekundy odhalit rozdíly v barvě povrchu. Potíž však nastane, má-li se v takovém případě zasáhnout při objevení chyby. Z technických důvodů není možné snadno zastavit extruzi střev a odstranit vadnou část výrobku. Jedinou možností zásahu je tedy nějakým způsobem označit vadnou část střeva za účelem její pozdější snadné identifikace. To lze provést kupříkladu nanesením nějakého zdravotně nezávadného barviva, které bude velmi dobře viditelné na střevu i po řásnění a přidání dalšího automatizovaného kontrolního místa, které bude mít za úkol vyřadit takto označené kusy.

Shrnutí:

- Výhody
  - Dobrá viditelnost vad
  - Snadná možnost programové segmentace vad
- Nevýhody
  - Velká rychlost pohybu střeva ztěžuje pořízení obrazu

– Není možné hned vyřadit vadnou část střeva

– Nutnost použít minimálně tři řádkové kamery

2.2 Možnosti umístění kontroly kvality před řásněním

Před řásněním, kdy se střevo odvíjí z kotoučů, jsou všechny optické vady na povrchu střeva dobře viditelné. Při převíjení má střevo v některých místech plošnou podobu a lze tak k jeho snímání použít dvou řádkových kamer a nalezené chyby snadno segmentovat. Problémem tohoto však řešení je stejně jako v předchozím případě obtížná možnost zásahu do výroby po zjištění chyby. Vyřazení nevyhovující části střeva by vyžadovalo zastavení převíjení, nebo složitou synchronizaci se zařízením, které by příslušnou vadnou část odstranilo až po zřásnění jako roubík.

Synchronizaci lze provést na základě znalosti průběhu rychlosti pohybu střeva v řásnícím zařízení. Označíme-li v_1 rychlost střeva na vstupu řásnícího stroje, $v_2 = kv_1$ rychlost pohybu roubíku na jeho výstupu a $T_r = f(v_1)$ čas průchodu řásnícím strojem, dále vzdálenost měření od vstupu řásnícího stroje l_1 a vzdálenost výstupu řásnění od místa vyřazení l_2, pak dobu T_c, za kterou se vada dostane na místo vyřazení, lze spočítat podle vzorce:

$$T_c = \frac{l_1}{v_1} + T_r(v_1) + \frac{l_2}{kv_1} \tag{1}$$

Dále lze určit, zda se vada bude vyskytovat uprostřed roubíku nebo na jeho kraji. To je třeba zohlednit, protože je nutno počítat s konečnou přesností délky střeva v roubíku a jiných významných veličin: Vada, která je podle výpočtu na konci současného roubíku může být ve skutečnosti na začátku roubíku následujícího.

Délka roubíku je dána intervalem oddělování v řásnícím stroji, který označíme T_i. Předpokládejme, že k prvnímu oddělení došlo v čase $t = 0$. Označme dobu, kdy dojde k nalezení vady t_x. Vada se pak objeví v místě vyřazení v čase $t = t_x + T_c$.

Koeficient $w \in \langle 0 : 1)$ určující polohu vadného místa na roubíku v poměru k jeho délce získáme podle vzorce[1]:

$$w = \frac{t_x + T_c - T_i \left[\frac{t_x + T_c}{T_i}\right]}{T_i} \tag{2}$$

Chceme-li tedy pro jistotu vyřadit dva roubíky, pakliže by se vada vyskytla v krajních deseti procentech délky, odstraníme předchozí roubík, pokud je $w \leq 0,1$ nebo další roubík, pokud je $w \geq 0,9$.

[1] Hranaté závorky ve vzorci označují funkci „celá část“.

Shrnutí:

- Výhody
  - Dobrá viditelnost všech vad
  - Snadná možnost programové segmentace vad
- Nevýhody
  - Velká rychlost pohybu střeva ztěžuje pořízení obrazu
  - Po zjištění vady problematický způsob zásahu za účelem její nápravy

2.3 Možnosti umístění kontroly kvality za řásněním

Po řásnění má výrobek podobu, ve které se dostane do ruky zpracovateli. Výrobce tedy zajímá především tato možnost. Hlavní nevýhodou je to, že část vad může být skryta uvnitř zřásněné struktury a není proto možné tyto vady odhalit. Tuto nevýhodu zmírňuje skutečnost, že největší vliv na spokojenost s výrobkem mají vady, které se objeví na povrchu roubíku.

Další nevýhodou je, že v zřásněné podobě obsahuje obraz střeva světlé a tmavé části, které brání použití jednoduchých metod rozpoznání vad.

Výhodou takového umístění je snadná možnost zásahu při zjištění vady. Najde-li automatizovaný systém kontroly kvality vadný roubík, lze jej vyřadit bez vlivu na zbytek výroby. Automatická kontrola kvality po řásnění je také bližší současné podobě kontroly, kdy pracovníci na výstupu řásnících strojů kontrolují každý výrobek manuálně.

Shrnutí:

- Výhody
  - Snadnější zásah při nálezu vady
  - Bližší současnému způsobu kontroly
  - Možnost odhalit i chyby které mohou vzniknout při řásnění
- Nevýhody
  - Část vad může být skryta uvnitř
  - Složitý způsob zpracování obrazu

Tato práce se bude dále zabývat kontrolou na výstupu řásnícího stroje, kde má střevo již podobu jednotlivých roubíků.

3 Úloha kontroly kvality

Kontrolu kvality na výstupu výroby pomocí analýzy obrazu můžeme shrnout do následujících tří částí:

1. Získání obrazové informace
2. Zpracování informace a rozhodnutí o kvalitě
3. Případný zásah za účelem nápravy

3.1 Získání obrazové informace

Jedná se o bod, který má zásadní vliv na celý proces. Je nutno získat o zkoumaném objektu takový obrazový materiál, který bude obsahovat všechny informace potřebné k rozhodnutí, zda výrobek vyhovuje, nebo zda bude vyřazen. U roubíku je navíc nutno vzít v úvahu vliv jeho válcovitého tvaru na podobu a rozsah obrazu a obsah detailu.

Důležitými body, které ovlivňují měření, jsou:

- Rozlišení kamery
- Osvětlení scény
- Použitý objektiv a optické filtry

Tímto tématem se dále zabývá část 5.

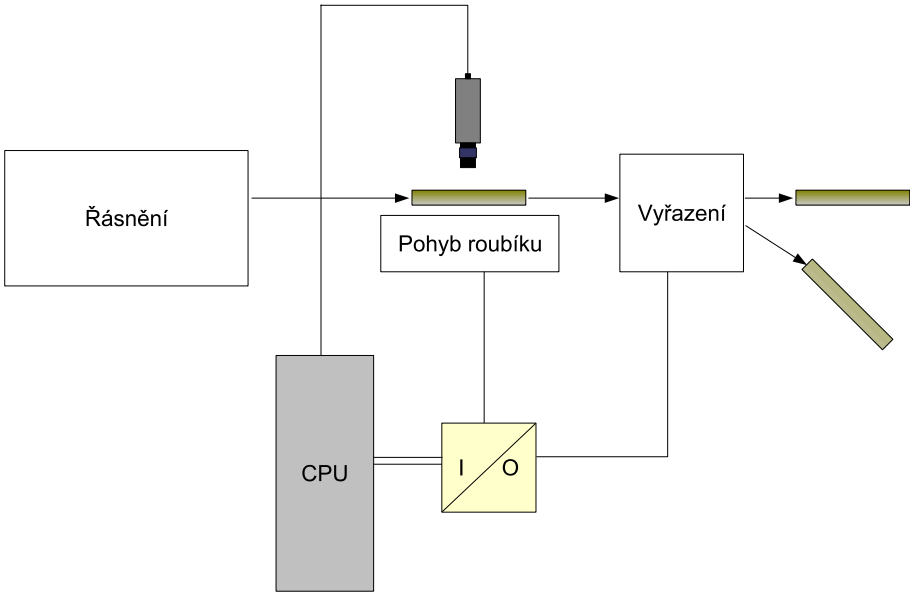
3.2 Zpracování informace a rozhodnutí o kvalitě

Nejprve je nutno v obraze najít zkoumaný objekt. Pro jeho nalezení, tedy segmentaci obrazu na objekt a okolí, vypočteme globální informace o obraze, především histogram. Poté využijeme tyto informace k extrahování obrazu, který obsahuje pouze povrch roubíku. Povrch následně prozkoumáme různými algoritmy zpracování obrazu za účelem nalezení a označení vad. Metody zpracování obrazové informace, které jsou použity, popisuje kapitola 4. Samotné postupy hledání vad jsou vysvětleny v části 6.

3.3 Možnosti zásahu v případě nalezení vady

Protože rozložení vzniku vad ve výrobě je náhodné a příčiny různé, není možné provádět zásahy do systému, které by měly za důsledek eliminaci jejich vzniku.

Reakcí na nalezení vady by tedy mělo být vyřazení vadného výrobku podobně, jak se tomu děje v současnosti. Jestliže systém kontroly bude mít podobu průmyslového počítače s kamerou připojenou přes zachytávací rozhraní jak ilustruje následující obrázek, lze připojit vstupně–výstupní rozhraní. Takové rozhraní umožní z vnějšku ovládat snímání kamerou kvůli synchronizaci s pohybem výrobků a zároveň digitálním výstupem dát signál zařízení, které vyřadí vadný roubík.



Obrázek 1: Ilustrace systému detekce vad

Za zmínku také stojí programovatelné logické automaty (PLC) umožňující zahrnout pořízení obrazu, jeho zpracování a signalizaci k zásahu do jednoho řešení.

4 Metody analýzy obrazu

4.1 Konvoluční masky aproximující derivaci

Filtrace konvoluční maskou je diskrétní konvoluce obrazové funkce $g(m, n)$, kde hodnota g vyjadřuje jas na daných souřadnicích, s konvoluční maskou h. Konvoluci v okolí O vyjadřuje rovnice 3.

$$f(x, y) = \sum_{(m,n)} \sum_{\in O} h(x - m, y - n)g(m, n) \tag{3}$$

Filtrace pomocí konvoluční masky tak umožňuje stanovit hodnotu jasu pixelu na základě jeho okolí O a koeficientů masky. Konvoluční masky lze využít pro filtraci obrazu za účelem vyhlazení, potlačení šumu, zvýraznění důležitých prvků obrazu aj. V této práci jsou použity především hranové operátory – konvoluční masky, které aproximují první nebo druhou derivaci a umožňují tak v obraze zdůraznit přechody mezi změnami jasu.

Robertsův operátor

Robertsův operátor je nejstarším a velmi jednoduchým způsobem aproximace první derivace. Používá okolí 2x2 reprezentativního pixelu. Jeho konvoluční masky jsou:

$$h1 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \qquad h2 = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \tag{4}$$

Velikost gradientu se počítá podle vzorce:

$$|g(x, y) - g(x + 1, y + 1)| + |g(x, y + 1) - g(x + 1, y)| \tag{5}$$

Nevýhodou Robertsova operátoru je velká citlivost na šum, protože okolí použité pro aproximaci je malé.

Laplaceův operátor

Laplaceův operátor je aproximací druhé derivace. Je invariantní vůči rotaci a udává pouze velikost hrany, ne její směr. Konvoluční masky Laplaceova operátoru mají tuto podobu:

$$h = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}, \qquad h = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{6}$$

Stejně jako Robertsův operátor, je i Laplaceův operátor velmi citlivý na šum. Při jeho použití také dochází ke zdvojení detekovaných hran u tenkých linií v obraze.

Sobelův operátor a operátor Prewittové

Tyto operátory detekují hrany v různých směrech na základě několika masek, které jsou vytvořeny rotací masky původní. Vybrána je ta maska, které odpovídá největší výsledná hodnota gradientu.

Sobelův operátor

$$h1 = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}, \; h2 = \begin{bmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ 2 & 1 & 0 \end{bmatrix}, \; h3 = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} ... \quad (7)$$

Operátor Prewittové

$$h1 = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}, \; h2 = \begin{bmatrix} 0 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 0 \end{bmatrix}, \; h3 = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} ... \quad (8)$$

Filtrace šumu průměrováním

Filtrace šumu průměrováním je jednoduchá a výpočetně nenáročná metoda filtrace šumu v obraze. Její postup spočívá v nahrazení každého pixelu hodnotou vypočtenou z jeho okolí velikosti $m \times m$. Tomu také odpovídá výsledek filtrace: Šum je potlačen, cenou za to je však degradace hran obrazu.

Filtrace šumu mediánem

Filtrace mediánem je velmi vhodná k potlačení šumu typu *pepř a sůl*, tedy náhodně rozmístěných světlých či tmavých pixelů. Takový šum se často vyskytuje při snímání obrazu kamerou, která používá CCD elementy. Filtr nahradí reprezentativní pixel mediánem vypočteným ze všech pixelů v okolí $m \times m$. Nevýhodou mediánové filtrace je rozostření detailů a při použití okolí větších rozměrů také výpočetní náročnost, protože je nutno pro každý pixel provést setřídění pole o délce m^2. Oproti filtraci průměrováním je však výsledný obraz ostřejší a šum lépe potlačen. Další výhodou je, že hodnota,

kterou se nahradí reprezentativní pixel, již v okolí pixelu existuje. Naproti tomu při použití průměrování dochází k nahrazení interpolovanými hodnotami, které se v obraze často vůbec nevyskytují.

4.2 Matematická morfologie

Matematická morfologie se používá především pro předzpracování obrázků, k filtraci šumu a zdůraznění struktury objektů. Jedná se o soubor operací s obrazem, které využívají popisu obrazu množinou bodů v prostoru ε^2 nebo, pro šedotónové operace, ε^3. Morfologické operace bývají nejčastěji realizovány jako relace obrazu (bodové množiny X) s menší bodovou množinou B, kterou nazýváme strukturní element.

Binární dilatace a eroze

Dilatace skládá body dvou množin pomocí vektorového součtu. Používá se k zaplnění děr a zálivů v objektech a zdůraznění malých objektů. Dilatace je definována vztahem:

$$X \oplus B = \{d \in \varepsilon^2 : d = x + b, x \in X, b \in B\} \tag{9}$$

Eroze je duální transformace k dilataci. Skládá dvě množiny s využitím rozdílu vektorů. Používá se k odstranění drobných objektů a bodového šumu, nebo k zvýraznění zálivů a děr. Lze ji také použít pro získání obrysů binárního obrazu. Eroze je definována vztahem:

$$X \ominus B = \{d \in \varepsilon^2 : d + b \in X, \forall b \in B\} \tag{10}$$

Dilatace a eroze zobecněné pro obrazy ve stupních šedi

Dilatace a eroze šedotónových množin je zobecněním binárních operací dilatace a eroze. V [1] je vysvětlen způsob tohoto zobecnění. Šedotónový obraz si lze představit jako funkci $z = f(x, y)$, kde z odpovídá jasové hodnotě na souřadnicích (x, y). Jedná se o množinu bodů v Euklidovském prostoru ε^3.

Pro množinu A v n-rozměrném Euklidovském prostoru, definuje [1] pojem *vršek*. Vršek množiny A je funkce definovaná na $(n-1)$-rozměrném definičním oboru. Pro každou $(n-1)$-tici je vršek nejvyšší hodnota zbylé poslední souřadnice množiny A.

Nechť $A \subseteq \varepsilon^n$ a nechť definiční obor $F = \{x \in \varepsilon^{n-1}$ pro některá $y \in \varepsilon, (x, y) \in A\}$. *Vršek* množiny A, označovaný $T[A]$, je zobrazením $F \to \varepsilon$ definovaným jako

$$T[A] = max\{y, (x, y) \in A\}. \tag{11}$$

Dalším důležitým pojmem je *stín* funkce f. Ve 3D si stín funkce f můžeme představit jako množinu sestávající z vršku f a celého prostoru pod ním. Formálně je stín definován takto:

Nechť $F \subseteq \varepsilon^n$ a $f : F \to \varepsilon$. *Stín* funkce f se označuje $U[f]$, $U[f] \subseteq F \times \varepsilon$ a je definovaný vztahem

$$U[f] = \{(x, y) \in F \times \varepsilon, y \leq f(x)\}. \tag{12}$$

Nyní můžeme definovat šedotónovou dilataci dvou množin jako vršek (binární) dilatace jejich stínů. Nechť $F \subseteq \varepsilon^n$, $f : F \to \varepsilon$ a $k : K \to \varepsilon$. Šedotónová dilatace $\oplus$ funkce f s funkcí k je definována takto:

$$f \oplus k \ = T\{U[f] \oplus U[k]\}. \tag{13}$$

Podobně šedotónová eroze $\ominus$ je vršek dilatace stínů příslušných množin:

$$f \ominus k \ = T\{U[f] \ominus U[k]\}. \tag{14}$$

Šedotónové otevření a uzavření

Operace *otevření* a *uzavření* jsou tvořeny kombinací eroze a dilatace. Tyto operace se používají k extrakci částí obrazu s definovaným tvarem a šedotónovou strukturou. Lze je využít pro hledání lokálních extrémů jasové funkce.

Šedotónové otevření $f \circ k$ je definováno takto:

$$f \circ k \ = (f \ominus k) \oplus k. \tag{15}$$

Podobně šedotónové uzavření $f \bullet k$:

$$f \bullet k \ = (f \oplus k) \ominus k. \tag{16}$$

Příkladem využití šedotónové morfologie je segmentace objektu na obrázku 4.2. Pro získání objektu byla na výchozí obrázek aplikována nejprve operace otevření se strukturním elementem velikosti 1×30, jehož všechny prvky jsou rovny 1. Dále bylo aplikováno uzavření se stejným strukturním elementem. Prahováním tohoto obrazu získáme výsledný motiv.

a) Původní obraz a obraz zpracovaný otevřením a uzavřením b).

c) Výsledná segmentace

Obrázek 2: Ukázka segmentace využitím šedotónové morfologie

Transformace vrchní část klobouku a spodní část klobouku

Mějme šedotónový obraz X a strukturní element K. Operací *vrchní část klobouku* nazýváme množinový rozdíl mezi otevřeným a původním obrazem, tedy $X \setminus (X \circ K)$.

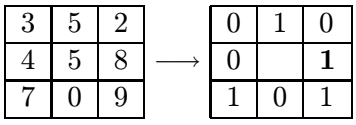
Podobně *spodní část klobouku* je definována jako množinový rozdíl mezi původním a uzavřeným obrazem: $(X \bullet K) \setminus X$

Tyto dvě operace se používají pro segmentaci objektů v šedotónovém obraze, kde se pomalu mění pozadí. Při vhodné volbě strukturního elementu lze při segmentaci dosáhnout různé citlivostí v různých směrech.

4.3 Operátor LBP

Operátor LBP (Local Binary Pattern) je nástroj umožňující popsat vzor (texturu) vektorem hodnot. Charakterizuje lokální vzory v textuře prahováním sousedních pixelů hodnotou prostředního pixelu. V obecné definici operátoru může mít rádius sousednosti a množství pixelů v něm libovolnou

hodnotu. Zde je bráno v úvahu osm sousedních pixelů – osmiokolí pixelu a rádius 1. Tento operátor je potom označen jako LBP8,1. Na obrázku 4.3 je příklad výpočtu neinterpolovaného LBP8,1. Okolí pixelu je prahováno hodnotou jeho jasu a výsledný binární vzor je reprezentován číslem, které vznikne seřazením binárních hodnot prahovaného vzoru za sebe. Rozdělení těchto čísel v textuře v podobě histogramu je použito k její identifikaci.



Binárně: 10100101
Decimálně: 165

Obrázek 3: LBP - vlevo původní okolí, vpravo prahované. Binární kód je čten v kladném směru.

K ohodnocení textury se pak použije rozložení LBP hodnot v obraze – tedy pro obraz v 256ti úrovních šedi vektor rozměru 256. Operátor LBP tak umožňuje pro účely rychlého rozpoznávání nahradit matici textury vektorem, a to velice efektivně. Charakteristikou LBP je necitlivost na globální změny úrovně jasu v důsledku prahování každého lokálního vzorku. Je však citlivý na rotaci; v [8] se lze dočíst o variantách LBP, které jsou k rotaci indiferentní. V této práci je však předpokladem správná orientace objektu při snímání obrazu, proto zde bude použita tato základní forma operátoru LBP.

4.4 Kohonenovy mapy (SOM)

Kohonenovy mapy nabízí alternativu k obvyklým metodám analýzy vzorů založeným na nervových sítích. Výpočetní obtížnost neparametrických třídících algoritmů, jako je například algoritmus k-NN (k-Nearest Neighbor) je lineární funkcí počtu trénovacích vzorků. To způsobuje problémy s efektivitou u mnoha aplikací vizuální kontroly.

Kohonenovy mapy jsou technikou vizualizace dat, která snižuje rozměr zobrazených dat použitím nervových sítí, které samy sebe uspořádávají. Tato technika řeší problém zobrazení mnohorozměrných dat v jednorozměrném nebo dvourozměrném prostoru, kterému člověk snadno porozumí. Způsob, kterým Kohonenovy mapy umožní zobrazit vícerozměrná data v jednom nebo dvou rozměrech, je seskupení podobných vektorů dohromady. To je vlastnost, kterou můžeme využít při klasifikaci textur (vzorů).

4.4.1 Princip Kohonenových map

Jak už bylo psáno výše, Kohonenova mapa je pole neuronů. Při procesu učení mapy se jednotlivé neurony přizpůsobují přicházejícím podnětům. Proces učení je soutěživý a neřízený - to znamená že není potřeba žádný tzv. učitel[1], který určuje správný výstup pro konkrétní vstup. V základní verzi Kohonenových map je aktivován pouze jeden uzel (vítěz) jako odezva na jeden vstup. Místa odezvy na podobné vstupní signály se postupem procesu učení uspořádají, vznikne jakýsi nelineární souřadný systém mapující vstupní hodnoty na síti neuronů.

Učení mapy

Vektory, které jsou vstupem pro Kohonenovu mapu, musí být normalizovány. To znamená, že každá sada vstupních hodnot (atributů) musí být vynásobena koeficientem tak, aby její absolutní hodnota byla rovna jedné. Tím lze předejít dominanci jednoho atributu.

Každý neuron mapy má počet vstupů shodný s počtem atributů použitých ke klasifikaci. Vstupy neuronu jsou váhové koeficienty w_i. Učení mapy pak znamená hledání neuronu, který nejlépe odpovídá vstupnímu vektoru - jeho váhové koeficienty jsou nejblíže vstupním atributům. Tento neuron je pak označován jako *vítězný neuron*. Všem neuronům uvnitř rádiusu r kolem vítězného neuronu jsou upraveny váhy podle vstupního vektoru. Velikost modifikace váhy je nepřímo úměrná vzdálenosti od vítězného neuronu. Rádius r se snižuje se vzrůstajícím počtem iterací.

Proces učení končí, jestliže je dosaženo požadované odchylky vah neuronů od vstupních dat, nebo je-li dosaženo předem určeného počtu iterací.

Existují dvě základní metody, jak určit podobnost mezi vstupním vektorem a neuronem. První metodou je skalární součin vektoru atributů a vektoru vah jednoho neuronu. Označíme-li k k-tý neuron v síti a N rozměr vektoru atributů, výstup neuronu v určíme podle rovnice:

$$v = \sum_{i=1}^{N} [x(i).w(i,k)] \tag{17}$$

Výsledkem skalárního součinu je projekce jednoho vektoru na druhý. Jestliže definujeme jednotkové vektory ve směrech $x(i)$ a $w(i)$, jejich skalární

[1] z anglického „supervisor“

součin bude roven kosinu úhlu mezi vektory x a w. To je jedním z důvodů, proč je nutné sady vstupních dat normalizovat. Výstup v roven jedné pak znamená, že vektory jsou rovnoběžné, tedy data jsou podobná - výstup roven nule znamená kolmost vektorů a naprostou neshodu vstupu s neuronem.

Druhou metodou určení podobnosti mezi vektory je výpočet Euklidovské vzdálenosti podle rovnice:

$$v = \sqrt{\sum_{i=1}^{N} [x(i) - w(i,k)]^2} \tag{18}$$

V tomto případě výstup roven nule znamená, že vektory jsou identické. Hodnota blízká dvojnásobku jejich normalizovaných absolutních hodnot bude znamenat, že jsou vektory naprosto odlišné.

Výpočet zopakujeme pro všechny neurony mapy a jako vítězný neuron označíme neuron s největším výstupem v případě použití skalárního součinu, nebo s nejmenším výstupem v případě použití Euklidovské vzdálenosti.

Proto, aby došlo k seskupení podobných neuronů, jsou váhy všech neuronů ve vybraném rádiusu r upraveny dle atributů vstupu. Nejvíce jsou modifikovány váhy vítězného neuronu; váhy ostatních neuronů se změní méně podle nepřímé úměry jejich topologické vzdálenosti od vítězného vektoru. Se vzrůstajícím počtem iterací se rádius zmenšuje, takže nakonec dojde ke změně vah pouze u vítězných neuronů. Pokud vítězí stále stejný neuron, je zmenšen koeficient na jeho výstupu (standardně roven jedné) za účelem snížení pravděpodobnosti výběru tohoto neuronu.

V závěru učení reprezentují váhy každého neuronu Kohonenovy mapy referenční data. Toho se využívá při pozdějším rozdělení do shluků a třídění pomocí mapy.

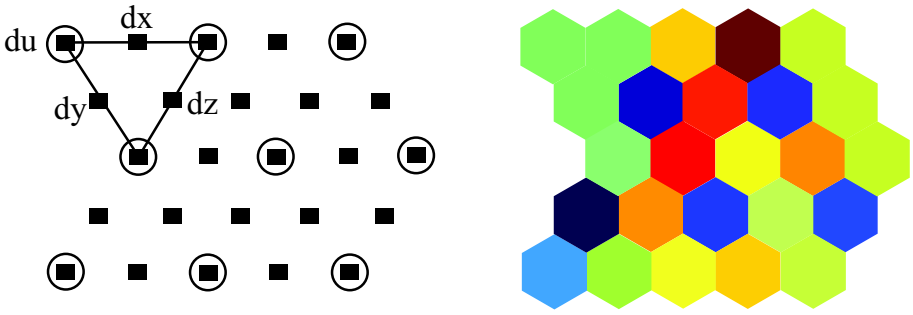
Klasifikace pomocí mapy

Kohonenova mapa, která projde procesem učení, je poté rozdělena na jednotlivé shluky, které reprezentují třídy vstupních dat. Pro rozdělení do shluků existují různé algoritmy, lze jej ovšem provést také ručně pomocí některého ze způsobů vizualizace Kohonenovy mapy (viz níže). Pro klasifikaci vzorku dat do jednoho ze shluků provedeme jednu iteraci jako v případě učení s tím rozdílem, že neměníme váhy neuronů; pouze nalezneme vítězný neuron. Vzorek pak přiřadíme do stejné třídy, ke které náleží vítězný neuron.

Vizualizace SOM metodou U-matice

Metoda U-matice (Unified distance matrix) umožňuje vizualizaci vícerozměrných dat v dvourozměrném prostoru s použitím Kohonenovy mapy jako zdroje dat. Toho je dosaženo využitím topologických relací mezi neurony, které se vytvoří procesem učení. Algoritmus vytvoří matici, jejíž každý člen je určen vzdáleností mezi dvěma sousedními neurony. Tak můžeme zobrazit vícerozměrnou matici dat pouze ve dvou rozměrech. Obrázek 4.4.1 ukazuje reprezentaci hexagonální Kohonenovy mapy rozměru 3x3 pomocí U-matice. Z této podoby můžeme zjistit relace mezi neurony a usuzovat na vstupní datovou strukturu. Vezmeme-li v úvahu, že každý neuron v mapě rozměru $m \times n$ má 256 vah, jak je tomu dále v této práci, je grafické znázornění takové matice obvyklými metodami jen těžko srozumitelné. Použitím U-matice je však možno přehledně zobrazit odchylky velikostí vektorů vah a vidět tak rozdělení mapy do jednotlivých shluků.

Vysoké hodnoty v U-matici reprezentují hranici mezi shluky a nízké hodnoty ukazují na velkou podobnost mezi neurony v dané oblasti - shluk.



Obrázek 4: Vizualizace Kohonenovy mapy pomocí U-matice.

4.4.2 Shrnutí algoritmu aplikace Kohonenovy mapy

1. **Inicializace dat:** Vektory zdrojových dat normujeme na absolutní hodnotu rovnou jedné. Pokud považujeme některý z atributů za významnější než ostatní, můžeme jeho absolutní hodnotu zvětšit. Tím bude více zohledněn při výpočtu vzdálenosti vektorů.

2. **Inicializace neuronů:** Neuron bude mít jeden váhový koeficient pro každý z atributů vstupních dat. Na začátku můžeme přiřadit vahám malé náhodné hodnoty, lineárně vrůstající hodnoty, nebo náhodně vybrané hodnoty ze vstupních dat. Zvolíme počáteční rádius r, popřípadě čítače, které počítají, kolikrát byl neuron zvolen jako vítězný.

3. **Výběr dat:** Vektory jsou vybírány v náhodném pořadí. Tím se zamezí nežádoucímu ovlivnění vývoje mapy výskytem podobných atributů za sebou. Počet iterací často bývá několikanásobně větší než počet vzorků; stejné vzorky vstoupí do procesu učení opakovaně.

4. **Určení vítězného neuronu:** Pomocí skalárního součinu (rovnice 17) nebo Euklidovské vzdálenosti (rovnice 18) projdeme celou Kohonenovu mapu a určíme, který z neuronů bude pro daný vzorek označen jako vítězný.

5. **Upravení váhových koeficientů v okolí vítězného neuronu:** Váhové koeficienty vítězného neuronu jsou upraveny podle rovnice 19, kde n je číslo iterace, k je index vítězného neuronu a j je j-tý atribut.

$$w_{n+1}(k,j) = w_n(k,j) + \eta(n)[x(j) - w_n(k,j)] \tag{19}$$

   Výpočet modifikace váhových koeficientů v okolí vítězného neuronu je dále rozšířen o funkci d, která zohlední vzdálenost daného neuronu od vítěze. Koeficient d je funkcí vzdálenosti a také hodnoty iterace n. Rovnice pro neuron uvnitř rádiusu s indexem m bude mít tuto podobu:

$$w_{n+1}(m,j) = w_n(m,j) + d(m,k,n)\eta(n)[x(j) - w_n(m,j)] \tag{20}$$

6. **Zmenšení poloměru** r**:** Zpočátku může být r tak velké, že změna ovlivní většinu neuronů celé mapy. S rostoucím počtem iterací se zmenšuje. Funkce $d(m,k,n)$ může být jakákoliv funkce, která klesá s rostoucí vzdáleností mezi neurony k a m a s rostoucím počtem iterací.

7. **Test ukončení algoritmu** Algoritmus končí, je-li dosaženo předem určeného maximálního počtu iterací, nebo pokud je celková RMS odchylka menší, než stanovená hodnota. Označíme-li M počet vzorků, ze kterých chceme RMS odchylku určit, $\mathbf{x}_i$ vektor vzorku a $\mathbf{w}_i$ vektor vah vítězného neuronu pro tento vzorek, RMS odchylku vypočteme takto:

$$RMS = \sqrt{\frac{1}{M}\sum_{i=1}^{M}[\mathbf{w}_i - \mathbf{x}_i]^2} \tag{21}$$

   Pokud není splněna žádná z podmínek ukončení algoritmu, postup pokračuje další iterací od bodu 3.

5 Získání obrazové informace

Získání obrazové informace stojí na začátku celého procesu, je to tedy jeho stěžejní část. Nastavením osvětlení, zvětšení a dalších faktorů, lze proces analýzy obrazu zjednodušit, ale také znemožnit.

5.1 Vybavení pro snímání obrazu

Vybavení můžeme rozdělit na 3 části:

- Kamera
- Objektiv a filtry
- Osvětlení

5.1.1 Kamera

Při výběru kamery musíme vzít v úvahu především velikost objektu, skutečnost zda je objekt v době snímání v pohybu a případnou rychlost, požadovaný detail a také to, zda je nutno snímat objekt barevně nebo zda vystačíme s odstíny šedi.

Předpokladem je, že při snímání roubíku nedochází k jeho pohybu, nebo je pohyb velmi pomalý ve srovnání s frekvencí kamery. Pak můžeme použít standardní průmyslovou kameru s televizní normou, která snímá obraz prokládaně s frekvencí 50 Hz pro půlsnímek a 25 Hz pro celý snímek. Pokud je v takovém případě objekt v pohybu, je nutno počítat s rozmazáním snímku, které je úměrné době expozice a rychlosti pohybu vůči ose kamery. Dále je nutno vzít v úvahu vliv prokládání snímků, který se v případě snímání pohybujícího se objektu projeví „roztřepením“ hran – rozdílem sudých a lichých řádků.

Pokud chceme snímat roubík v pohybu jedním směrem (např. na běžícím pásu), lepším řešením je použití řádkové kamery. Tyto kamery snímají jednotlivé řádky obrazu s velkou frekvencí, kterou je možno nastavit ve vhodném rozsahu tak, aby odpovídala rychlosti pohybu zkoumaného objektu. Často je součástí kamery i digitální rozhraní a software umožňující rekonstrukci obrazu, takže pro další zpracování má obraz stejnou podobu jako při snímání plošnou kamerou.

Rozlišení kamery zvolíme podle velikosti zkoumaných objektů a požadovaného rozměru nejmenšího detailu. Nejmenší velikost zachytitelného detailu v jednom směru můžeme vypočíst jako poměr rozměru objektu a rozlišení v daném směru, tedy $d = \frac{l}{res}$ za předpokladu, že objekt zabírá maximální

možnou šířku snímače kamery. Do vzorce je třeba dosadit rozlišení snímacího elementu kamery; to je zpravidla nižší než rozlišení na výstupu kamery. Dosažení dobré citlivosti také závisí na ploše snímacího elementu. Podrobnější popis lze získat v [2].

Při testech v laboratorních podmínkách byla použita černobílá plošná kamera s TV normou, 1/4" CCD snímačem a výstupním rozlišením podle CCIR normy 768×576. Při rozměru roubíku 15 cm je teoretická nejmenší velikost zachytitelného detailu přibližně 0,2 mm.

5.1.2 Objektiv a optické filtry

Volba objektivu závisí na velikosti snímače kamery, vzdálenosti kamery a velikosti objektu. Vztahy pro výpočet vhodné ohniskové vzdálenosti lze najít v [2]. Při testech v laboratoři byl použit 12 mm objektiv s pevnou ohniskovou vzdáleností a 6–15 mm objektiv s proměnlivou ohniskovou vzdáleností.

Při prvních pokusech snímání roubíku kamerou docházelo k problémům způsobeným odlesky. Pro odstranění takto vzniklého přeexponování byl před objektiv umístěn polarizační filtr, který byl natočen tak, aby maximálně bránil v průchodu světlu odraženému od objektů.

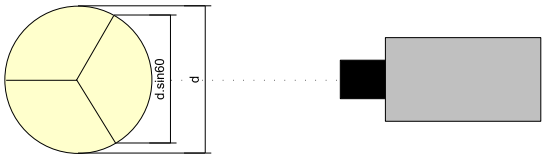
5.1.3 Osvětlení

Pro volbu osvětlení je první důležitou věcí odstínění okolního světla. Venkovní osvětlení se různě mění s časem, dochází k pohybu osob nebo strojů kolem kontrolního místa a tím může dojít k ovlivnění měření. Zejména dále popsaný způsob efektivní segmentace roubíku a pozadí je citlivý na výskyt světlých objektů v zorném poli kamery. Řešením tohoto problému může být použití monochromatického zdroje světla v kombinaci s barevným filtrem před objektivem, který maximálně propouští zvolenou vlnovou délku světla a má strmou charakteristiku.

Vzhledem k válcovému tvaru roubíku je žádoucí zvolit několik nezávislých světelných zdrojů nebo plošný světelný zdroj tak, aby byl povrch rovnoměrně osvícen z několika stran. Tím předejdeme vytvoření velmi pozvolného tmavého přechodu na jedné straně, který by zabránil správné segmentaci.

5.2 Vliv tvaru roubíku

Aby mohl být při kontrole kvality obsažen celý povrch roubíku, je třeba pořídit minimálně tři obrázky pro každý roubík. Tento případ ilustruje následující obrázek.

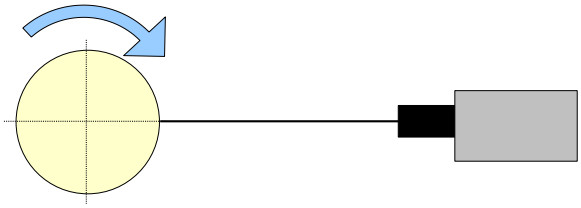


Obrázek 5: Pokrytí povrchu při třech záběrech na roubík

V případě, že se objeví vada na hranici průniku dvou snímků, které odpovídá úhel 60°, se však snižuje pravděpodobnost její odhalení. Malý výsek povrchu válce v úrovni odpovídající tomuto úhlu si lze představit jako šikmou rovinnou plochu velikosti S, kterou kamera snímá pod úhlem 30°. Bez uvažování vlivu zkreslení objektivu je tedy plocha skvrny, kterou vidí kamera S' dána vztahem $S' = S \sin 30°$, a redukuje se tak na 50 % skutečné plochy. Zkreslení způsobené perspektivou tuto plochu dále zmenšuje.

Obzvláště u roubíků kratšího typu se objevují na krajích záběru zubaté okraje způsobené řásněním. Tyto tmavé okraje je nutno po segmentaci oříznout, jinak mohou mít vliv na výsledek detekce. Jejich oříznutí však také snižuje pravděpodobnost odhalení vady, která se v době snímání obrazu vyskytne na hranici průniku dvou snímků.

Z těchto důvodů se v této práci doporučují v případě použití plošné kamery čtyři záběry pro každý roubík. Tím se o třetinu zvýší výpočetní a časová náročnost měření, avšak také se významně zvýší pravděpodobnost odhalení chyby.



Obrázek 6: Snímání povrchu řádkovou kamerou

Další možností je použití řádkové kamery, bude-li například roubík umístěn na otočném trnu, jak je znázorněno na obrázku 5.2. To by umožnilo zachytit celý povrch na jednom obrázku.

6 Realizace hledání nečistot v obraze

6.1 Nalezení roubíku v obraze

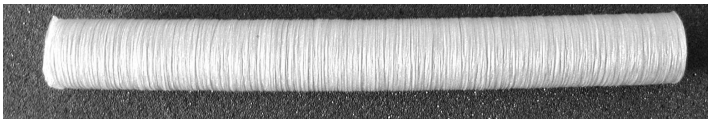
Základní úlohou při hledání vad na obraze roubíku je oddělení samotného objektu roubíku od pozadí obrazu (segmentace). Lze totiž předpokládat, že roubík nebude vůči kameře vždy ve stejné poloze a že se může měnit jeho délka. Pro segmentaci využijeme rozdílu jasů pixelů, které přísluší objektu od jasu pixelů příslušícímu pozadí. Segmentaci lze realizovat takto:

1. Přiblížení obrazu
2. Výpočet prahu pro rozdělení obrazu na objekt a pozadí
3. Prahování
4. Filtrace

Obrázek 7: Roubík

6.1.1 Přiblížení obrazu

Abychom zbytečně nezatěžovali hardware na kterém poběží výpočet, před prováděním náročnějších operací zmenšíme výřez obrazu na minimum. Využijeme znalosti obrazu - předpokládáme světlý objekt na tmavém pozadí - můžeme rozdělit obraz na pravidelné segmenty (matice) a vyřadit ty segmenty, kde se nachází nízký počet světlých pixelů.

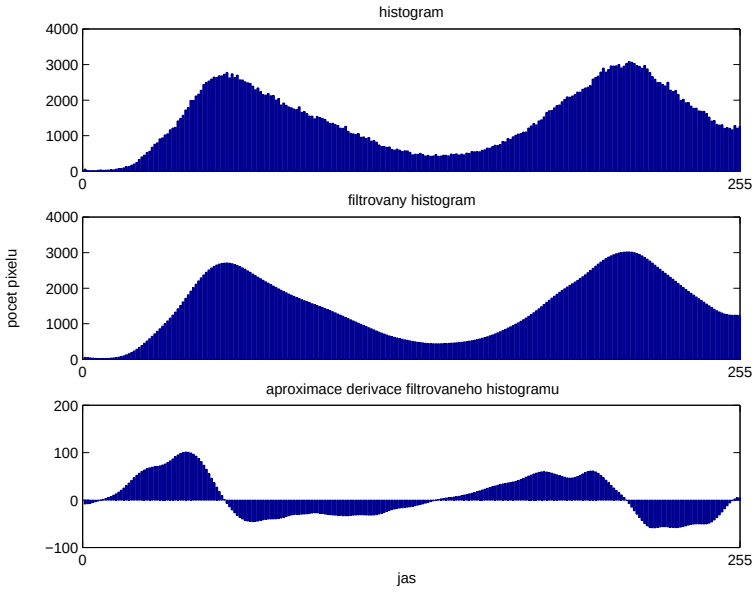


Obrázek 8: Přiblížení obrazu

6.1.2 Výpočet prahu

Pro oddělení objektu od pozadí je třeba určit práh. Jedná se o celé číslo 0-255, které reprezentuje jas pixelu určující hranici mezi pixely, které jsou považovány za objekt a pixely, které jsou považovány za pozadí. Nejprve je z celého obrázku vypočítán histogram jasových hodnot. Tento histogram ukazuje součty pixelů obrazu v jednotlivých jasech.

V histogramu hledáme hodnotu, která nejlépe oddělí tmavé a světlé objekty. V tomto případě jsou zde dvě velké skupiny jasových hodnot a lokální minimum mezi nimi je hledaným prahem.



Obrázek 9: Histogram

Postup nalezení lokálního minima v histogramu:

Histogram je třeba nejprve filtrovat. K tomu použijeme vhodný diskrétní filtr typu FIR (Finite Impulse Response – konečná impulsní charakteristika) nebo IIR (Infinite Impulse Response – nekonečná impulsní charakteristika).

Na obrázku 6.1.2 uprostřed je zobrazen histogram filtrovaný Butterworth filtrem 4. řádu. Posloupnost byla filtrována dvakrát – jednou zleva, podruhé zprava (příkaz MatLabu „filtfilt“). Tím se předejde deformaci grafu vzniklé zpožděním filtru.

Dále histogram za účelem nalezení lokálního minima derivujeme. Z průběhu derivace histogramu snadno určíme lokální extrémy jako body kde graf prochází nulou a vybereme z nich hledané lokální minimum. Zjištěná hodnota prahu podle histogramu z obr. 6.1.2 je 137.

6.1.3 Prahování

Prahování je druh přechodu z obrázku ve stupních šedi na obrázek binární. Postupně se projde celý obrázek a hodnoty menší než práh jsou nahrazeny hodnotami 0, hodnoty větší nebo rovny prahu jsou nahrazeny hodnotami nenulovými (např. 255 pro největší jas při zobrazení ve stupních šedi).

6.1.4 Filtrace šumu

K dokončení separace objektů od pozadí obrazu je nutné odfiltrovat náhodný šum v podobě světlých bodů, které se dostaly do výsledného obrazu, avšak nepatří k hledanému objektu. V práci byly použity tyto dva způsoby filtrace:

- několikanásobné aplikování eroze a dilatace
- vybrání vlastního objektu hledáním sousednosti pixelů.

Filtrace použitím eroze a dilatace

Použitím eroze dojde k eliminaci malých světlých objektů v obraze tvořených zejména jednotlivými pixely a jejich malými shluky. Pokud erozi aplikujeme několikanásobně, odstraníme i objekty větší – záleží ovšem také na tvaru použité masky. Vedlejším efektem eroze je zmenšení velkého objektu, který nás zajímá. To napravíme použitím dilatace se stejnou maskou a se stejným počtem opakování. Tím odfiltrujeme šum a zůstane pouze jediný objekt – shluk pixelů, který reprezentuje námi snímaný roubík.

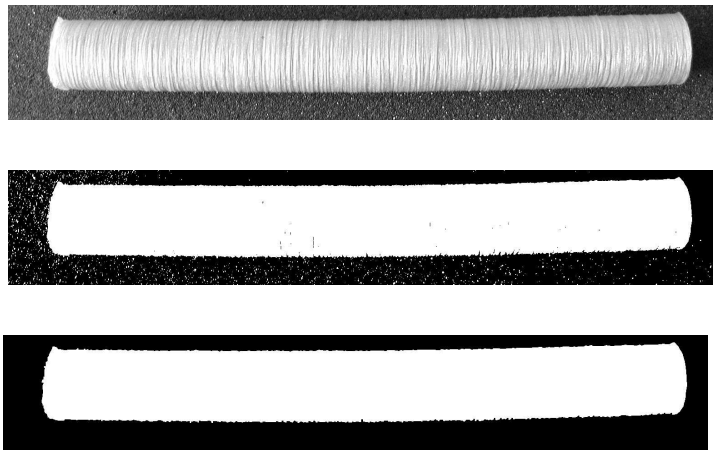
Filtrace použitím vybrání hledáním sousedních pixelů

Tento způsob používá příkaz bwselect programu MatLab. Vstupem algoritmu jsou souřadnice pixelu s hodnotou "1", který bezpečně patří k požadovanému objektu. Algoritmus začne s tímto pixelem a označí všechny přilehlé pixely s hodnotou "1" za součásti objektu. Stejný postup se dále

použije i na tyto pixely. Vybraná oblast ze zvětšuje, až dojde k výběru celého hledaného objektu.

Ukázka segmentace roubíku

Následující obrázek zobrazuje tři postupné kroky segmentace.



Obrázek 10: Nalezení roubíku v obraze: zoom, prahování, filtrace.

6.2 Realizace rychlého nalezení roubíku

Při použití vyhledávacího algoritmu v reálném čase jsou kladeny velké nároky na rychlost výpočtu. Proto byl postup realizován v jazyce C++ a optimalizován pro rychlost.

Rychlé nalezení souřadnic roubíku

1. výpočet histogramu
2. filtrace histogramu pomocí FFT
3. prahování
4. nalezení souřadnic

6.2.1 Rychlá filtrace histogramu pomocí FFT

Pro filtraci byl zvolen filtr typu FIR navržený pomocí MatLabu. Jedná se o dolní propust řádu 30 se zlomovou frekvencí $W_n = 0.05 f_s$. Pro rychlý

výpočet filtrovaného histogramu využijeme konvoluci a její výpočet pomocí diskrétní Fourierovy transformace. Průchod signálu $u(t)$ filtrem s impulsní přechodovou charakteristikou $g(t)$ (viz obrázek 11) je dán rovnicí 22.

$$y(t) = u(t) * g(t) = \int_0^t u(t-\tau)g(\tau)d\tau \tag{22}$$

Konvoluci $u(t) * g(t)$ získáme součinem příslušných Fourierových obrazů $U(f).G(f)$ a následnou inverzní Fourierovou transformací.

Postup je tedy následující:

1. Výpočet DFT histogramu pomocí knihovny FFTW
2. Součin DFT histogramu s předem vypočtenými hodnotami DFT filtru
3. Inverzní DFT tohoto součinu.

Konvoluce pomocí FFT je rychlejší než konvoluce v časové oblasti vzhledem k délce vektorů – histogram má délku 256 a impulzní přechodová charakteristika použitého filtru má 31 hodnot. Její průběh ukazuje obrázek 11. Další výhodou konvoluce ve frekvenční oblasti je nezávislost náročnosti výpočtu na počtu průchodů filtrem. Náročnost výpočtu je dále nezávislá na délce vektoru impulsní charakteristiky filtru, je-li tato menší než délka filtrovaného signálu.

Signál se nakonec diskrétně derivuje za účelem nalezení lokálních extrémů. Hodnota zjištěného prahu odděluje objekt od pozadí a použije se při prahování. Výsledkem je segmentovaný obraz, viz obrázek 6.1.4 uprostřed.

6.2.2 Nalezení souřadnic objektu

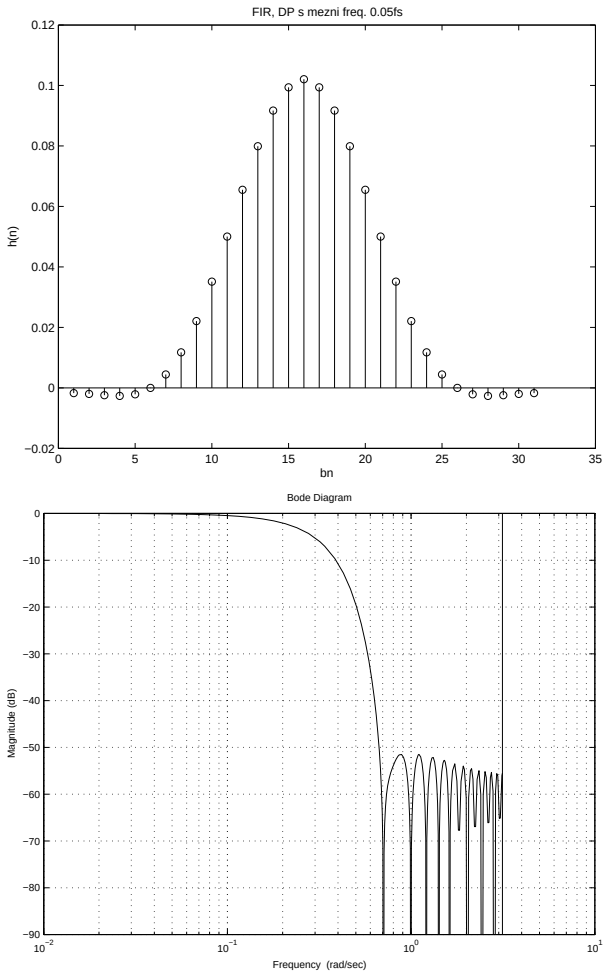
Vstupem pro další operace rozpoznávání povrchů je jen část obrazu, která přísluší objektu. Snahou je získat souřadnice co možná největšího obdélníku, který je celý obsažen uvnitř hranice objektu. Jednoduchým a rychlým postupem je vypočtení a následná analýza sloupcových a řádkových součtů v obraze. Takto vypočteme vektor sloupcových a řádkových součtů, jestliže MxN jsou rozměry obrázku a $p(i,j)$ je jasová hodnota na souřadnicích $[i,j]$.

$$s(i) = \sum_{j=0}^{N-1} p(i,j) \qquad s(j) = \sum_{i=0}^{M-1} p(i,j) \tag{23}$$

Analýzou skokových změn průběhů $s(i)$ a $s(j)$ získáme hledané souřadnice obdélníku, který je celý součástí objektu roubíku. Není nutné provádět výpočetně náročnou filtraci šumu. Předpokladem dobrého výsledku tohoto postupu je orientace objektu shodná s jednou z os obrazu.

6.2.3 Softwarové řešení segmentace v C++

Pro nasazení systému v praxi je velmi důležitá rychlost, s jakou je počítač schopen provést algoritmus zpracování. Proto zde byl kladen důraz na efektivitu algoritmu. Protože filtrace zde popsané jsou výpočetně náročné operace, v rámci této práce byl vytvořen program provádějící segmentaci v jazyce C++ s využitím jeho optimalizací.



Obrázek 11: Impulzní přechodová charakteristika použitého filtru a jeho frekvenční charakteristika.

Obrázek je v paměti reprezentován jako třída, která se skládá z obrazových dat a rozměrů. Data jsou uložena v podobě jednorozměrného pole typu int. To umožňuje velice rychlý sekvenční přístup k datům:

```
class  Picture
{
public:
   int*  data;
   int sizeX;
   int sizeY;
   //...metody
};
```

Jádrem programu je funkce `AutoTreshold`, která vypočítá automatickou segmentaci objektu od pozadí na základě průběhu histogramu. Funkce využívá dalších jednoduchých funkcí pro práci s obrazem – `Treshold`, `Histogram` a `DoFilter` (viz zdrojový kód v příloze). Všechny funkce jsou realizovány efektivně pomocí pointerů. Příkladem je funkce pro výpočet histogramu obrazu:

```
int*    ImageFunc::Histogram( Picture* pic )
{
   // vytvoreni a inicializace pole funkci memset
   int* hist = InitArray( HIST_LENGTH );
   int* p = pic->data;
   int* q = pic->data + pic->sizeX * pic->sizeY;
   for( ; p<q; ++p)
   {
      ++hist[*p];
   }
   return hist;
}
```

Zde je s výhodou použita reprezentace obrázku jednorozměrným polem – není třeba použít vnořené cykly for, celý obrázek se projde jedním cyklem. Pointer p ukazuje na aktuální pixel, pointer q je ukončovací a ukazuje na konec obrázku.

Pro filtraci histogramu pomocí FFT byly použity volně dostupné knihovny FFTW, které jsou psány v jazyce C a nabízí jednu z nejefektivnějších dostupných metod výpočtu rychlé diskrétní Fourierovy transformace. Metoda `DoFilter` volá FFTW funkci `fftw_execute_dft_r2c` pro výpočet dopředné transformace a `fftw_execute_dft_c2r` pro výpočet inverzní transformace.

Označíme-li $X(i)$ obraz vstupu, $F(i)$ obraz váhové funkce filtru a $\overline{F}(i)$ jeho komplexně sdružené číslo, pak FFT obraz dvojitě filtrovaného průběhu

$Y(i)$, který není posunut v čase, získáme takto:

$$Y(i) = F(i)\overline{F}(i)X(i) \tag{24}$$

Tomuto odpovídá následující kód v C++:

```
#define re(x) x[0]
#define im(x) x[1]
...
double fc;
for( int i=0; i<length; i++ )
{
   fc = re(filtr[i])*re(filtr[i]) + im(filtr[i])*im(filtr[i]);
   re(vystup[i]) *=  fc;
   im(vystup[i]) *=  fc;
}
```

Rychlost celé segmentace včetně následovného určení souřadnic obdélníku textury roubíku byla testována na PC s procesorem Athlon XP 1700+ – proces trval 21 ms pro obrázek o rozměrech 640×480 a filtr o délce 256. Taková rychlost plně postačuje potřebám této práce.

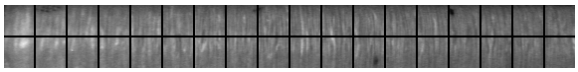
6.3 Vyhodnocení předzpracovaných dat

Dalším úkolem je nalezení nepravidelností v povrchu zkoumaného roubíku. První metodou, která k tomuto účelu bude využita, je klasifikace textur pomocí nervové sítě v podobě Kohonenovy mapy.

Jak je uvedeno v teoretické části této práce, Kohonenovy mapy (také SOM) umožňují třídění vícerozměrných datových vektorů a jejich zobrazení v jednom nebo dvou rozměrech – tedy v podobě, ve které je člověk snadno vizuálně zpracuje. Učením Kohonenovy mapy nejprve vytvoříme shluky (clustery), které reprezentují jednotlivé třídy, do kterých je možno zařadit vstupní data. Každý shluk reprezentuje vstupní vektor s nějakými vlastnostmi. Vzhledem k tomu, že učení Kohonenovy mapy probíhá bez učitele, nelze předem určit, jak bude vypadat výsledné rozdělení do shluků. Význam jednotlivých shluků lze určit experimentálně, nebo před učením mapy označit vstupní vektory a porovnat je s výsledky po skončení procesu.

6.3.1 Hledání vad pomocí Kohonenových map

Oblast zájmu rozdělíme na malé segmenty, které analyzujeme zvlášť. Jednotlivé segmenty jsou zpracovány mapou, která je předem naučena zpracováním mnoha podobných vzorků. Neporovnáváme však samotné segmenty textury. Z každého segmentu vypočteme vektor hodnot, který jej z hlediska určení povrchových nepravidelností dostatečně charakterizuje.



Obrázek 12: Rozdělení textury na segmenty

Charakterizace segmentu obrazu jednorozměrným vektorem

Pro porovnání metodou Kohonenových map je třeba místo samotných obrázků použít jednorozměrné vektory hodnot, které dostatečně reprezentují vlastnosti vzorů, podle kterých byly vytvořeny. Pro tyto účely bylo v rámci práce vytvořeno několik zkušebních programů pro zpracování obrazových vzorků a výpočet a zobrazení Kohonenových map.

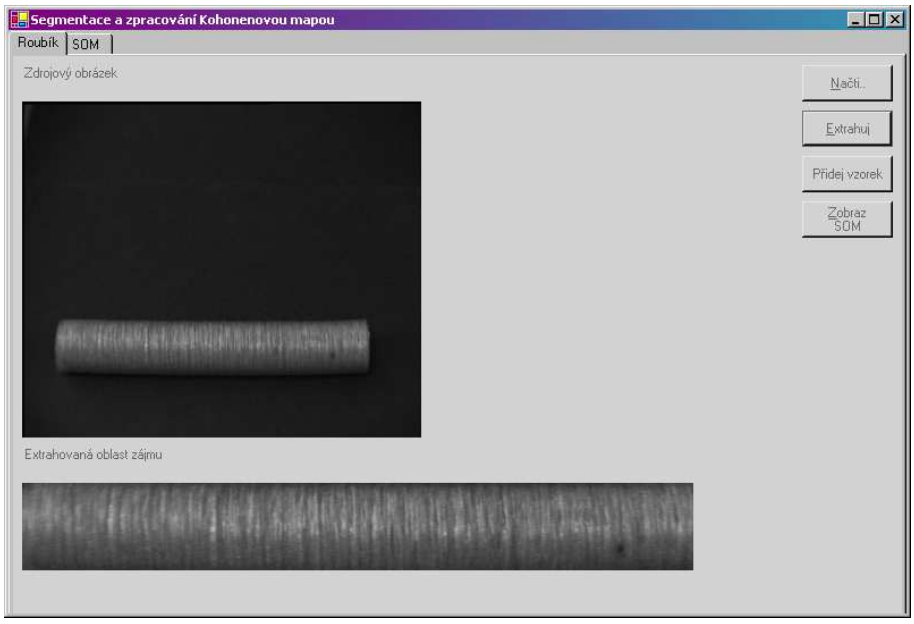
Použití algoritmu LBP

Algoritmus LBP (Local Binary Pattern) hledá v obrázku opakování malých vzorů rozměru 3x3 pixely. Každý z těchto malých vzorů je reprezentován osmibitovým číslem (hodnoty 0-255). Z celého obrázku je vytvořen histogram výskytu hodnot. LBP tedy reprezentuje obrázek vektorem celých čísel o délce 256.

Příprava dat pro zpracování

Pro testování funkce Kohonenových map bylo zapotřebí vytvořit program, který bude provádět segmentaci, jak je popsána v části 6.1, segmentovaný vzorek rozdělí na malé matice, které zpracuje pomocí LBP a výsledné vektory uloží do textového souboru jako vstup pro učení Kohonenovy mapy.

Uživatelské rozhraní programu ukazuje obrázek 6.3.1. V horní části okna programu je zobrazen originální obrázek, jak jej zachytila kamera. Pod ním je zobrazena extrahovaná část. Po kliknutí na tlačítko "Přidej vzorek" je tento segmentovaný obrázek rozdělen na matice o předem nastavené velikosti a každou z nich zpracuje a uloží.



Obrázek 13: Program pro segmentaci a zpracování obrazových dat z roubíků.

Pro účely učení map byl nasnímán soubor vzorků znečištěných tmavými skvrnami různých tvarů a velikostí. Soubor nasnímaných vzorků byl poté v programu zpracován a byly vybrány reprezentativní vzorky obrázků.

Učení mapy

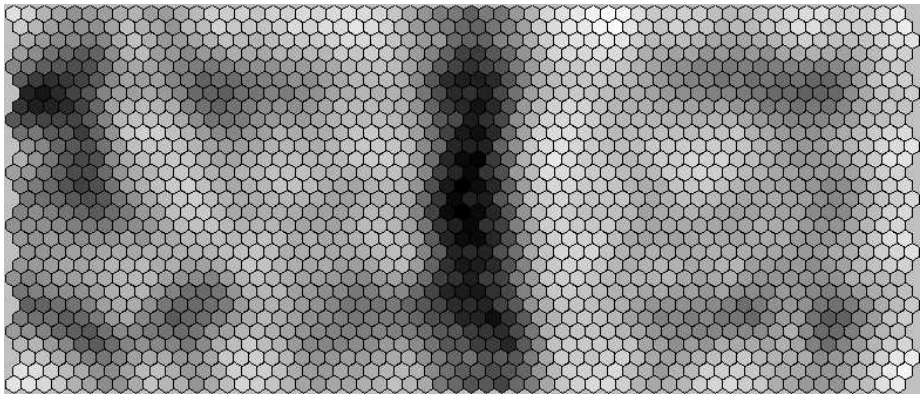
Program `som_training`,který vytvoří SOM mapu na základě vstupních dat z výše získaných matic, byl vytvořen v C++ za pomoci volně dostupných zdrojových kódů v jazyce C `som_pak`[1]. Před spuštěním učení mapy se musí nastavit potřebné parametry. Především se jedná o rozměr datového vektoru, požadované rozměry mapy a počet iterací. Dalšími parametry jsou hodnoty ovlivňující počáteční koeficient alfa (míru přizpůsobení vítězného uzlu) a počáteční rádius, který určuje počet přizpůsobených uzlů.

Rozměr datových vektorů musí být stejný jako rozměr vstupního vektoru – v tomto případě se jedná o histogram osmibitového LBP, rozměr d je tedy roven $2^8 = 256$. Pro velikosti map existují různá pravidla, která berou v úvahu především počet vstupních vzorků. V této práci je vyzkoušeno několik různých velikostí mapy. Počet iterací učícího algoritmu by měl překračovat několikanásobek počtu vstupních vektorů. Učení probíhá ve dvou fázích - v první fázi je použit menší počet iterací a větší koeficienty pro při-

[1] Zdrojové kódy byly získány z www.cis.hut.fi/research/som_pak/

způsobení uzlů. Druhá fáze slouží k doladění výsledků, použije proto větší počet iterací a menší koeficienty.

Výsledek zpracování program uloží do textového souboru. Ten obsahuje všechna data potřebná k zobrazení Kohonenovy mapy a použití ke klasifikaci dat. Příklad zobrazení výsledné mapy v podobě U-matice ukazuje obrázek 6.3.1.



Obrázek 14: Kohonenova mapa

Klasifikace pomocí mapy

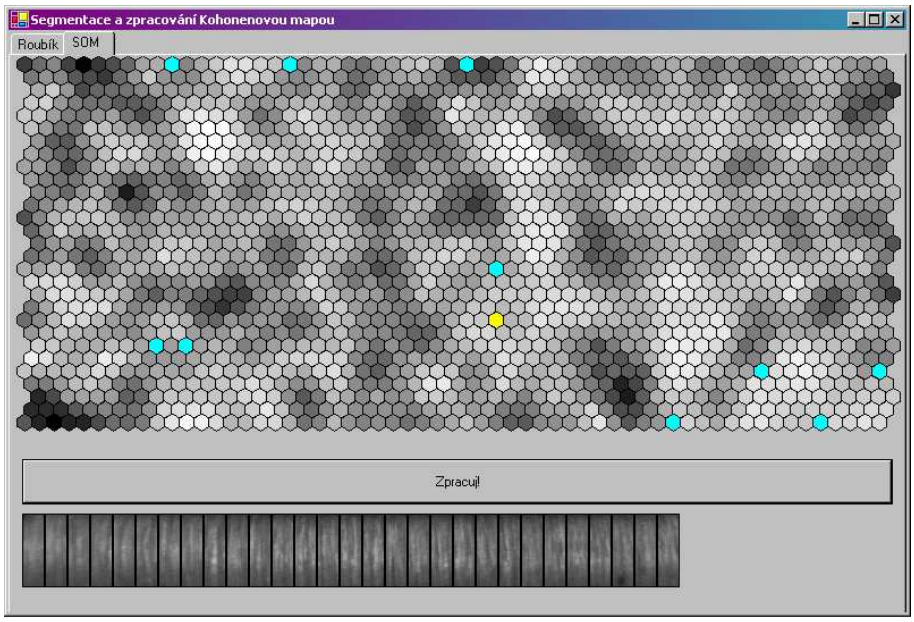
Program sloužící k segmentaci roubíku byl rozšířen o zobrazení SOM pomocí metody U-matice a klasifikaci jednotlivých vzorků a částí vzorků. Pro tyto účely bylo rovněž využito zdrojových kódů `som_pak`.Program používá jako vstup datový soubor, který je výstupem ze `som_training`.

Obsažená mapa je zobrazena metodou U-matice v okně programu, jak ukazuje obrázek 6.3.1.

Pod obrázkem mapy je zobrazen zkoumaný vzorek. Ten je rodělen na samostatně zpracované matice. Každá z těchto matic je klasifikována Kohonenovou mapou a příslušné uzly mapy jsou na obrázku mapy barevně odlišeny. Tak je možno zkoumat výsledek klasifikace. Obrázek vzorku ve spodní části okna je interaktivní – po nastavení ukazatele na konkrétní matici program označí uzel mapy, ke kterému matice patří.

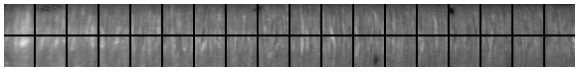
Analýza shluků

Podkladem pro mapu zobrazenou na obrázku 6.3.1 byly vzorky rozdělené do matic velikosti 32×32 tak, jak ukazuje obrázek 6.3.1. Při analýze mapy



Obrázek 15: Program pro segmentaci a zpracování obrazových dat z roubíků.

byl zaznamenán vznik dvou velkých shluků. Tyto shluky jsou reprezentovány světlými odstíny, charakterizujícími podobnost datových vektorů, a odděleny tmavými odstíny, které charakterizují značné rozdíly mezi vektory. Použitím této mapy je tedy možno rozdělit vzorky do dvou skupin.

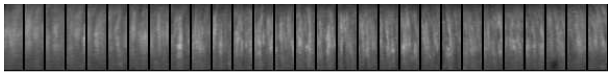


Obrázek 16: Rozdělení textury na segmenty 32×32

Přiřazením matic ke shlukům v programu však zjistíme, že spíše než obsah tmavých skvrn, které charakterizují vadu, má na umístění do shluku vliv poloha matice ve vzorku: Všechny matice z horní části vzorku patří do shluku vpravo a všechny dolní matice přísluší k levému shluku. Matice obsahující tmavé skvrny se vyskytují v obou shlucích a to na různých souřadnicích, které spolu nijak nesouvisí.

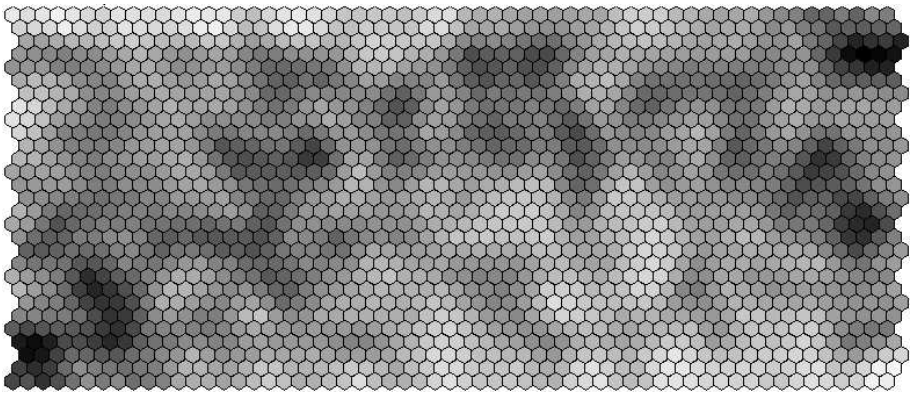
Při dalším pokusu byl proto vzorek rozdělen na matice obdélníkového tvaru, jak ukazuje obrázek.

Výsledná mapa je zobrazena na obrázku 6.3.1. Nedošlo zde k žádnému významnému rozdělení na shluky a ani při experimentování se změnami vstupních parametrů a velikosti mapy se nepodařilo získat uspokojivý výsle-



Obrázek 17: Rozdělení textury na segmenty 20×82

dek, který by byl použitelný ke klasifikaci vstupních dat za účelem kontroly kvality.



Obrázek 18: Kohonenova mapa při rozdělení na matice 20×82

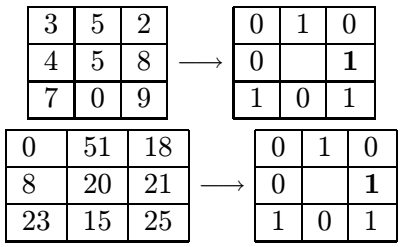
Tyto výsledky ukazují na to, že vzniklé Kohonenovy mapy nelze použít pro oddělení vadných objektů od objektů vyhovujících. Struktura samotného objektu má totiž na charakterizaci větší vliv, než přítomnost či absence tmavých skvrn.

Rozbor problému

Důležitou skutečností je, že při učení Kohonenovy mapy a pozdější klasifikaci nejsou použity přímo zkoumané části textury, ale jednorozměrné vektory LBP histogramů, kterými jsou tyto zkoumané části nahrazeny.

Z definice algoritmu LBP zjistíme, že číslo, které reprezentuje lokální vzor 3×3 kolem reprezentativního pixelu, je vytvořeno na základě prahovaných hodnot pixelů uvnitř lokálního vzoru. Hodnota každého pixelu je nahrazena hodnotou 0 nebo 1 podle toho, zda je větší, než hodnota reprezentativního pixelu. Z toho vyplývá, že při tomto procesu se ztrácí nejen informace o absolutní hodnotě jasu, ale také informace o velikosti rozdílu jasů uvnitř lokálního vzoru. Tento problém ilustruje obrázek 6.3.1.

Problém je tedy v použití operátoru LBP pro charakterizaci textury. LBP zdůrazní vliv změn tvarů ve vzoru a potlačí jasové vlivy. Umožňuje tak rozpoznávat různé vzory, které se liší především strukturou. Struktura roubíku obsahuje v různých místech nepravidelnosti a tvary vad na roubíku



Obrázek 19: LBP - stejný výsledek pro dva velmi odlišné lokální vzory.

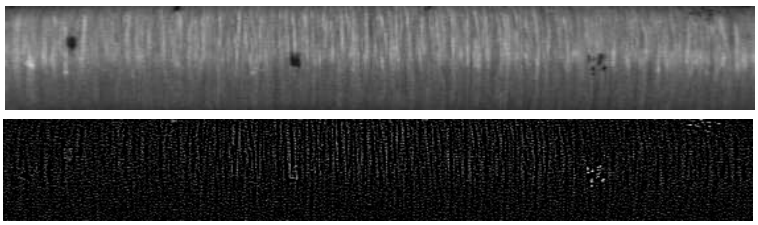
jsou také nepravidelné. Ve vektorech, které vzniknou použitím LBP se proto nevyskytuje žádná informace, kterou by bylo možno použít ke klasifikaci. Lokální poklesy jasu, které jsou příznakem tmavých skvrn na roubíku, se tak na výsledných mapách projeví jen minimálně.

Řešením tedy může být nahrazení LBP jiným algoritmem, který bude poskytovat informaci o textuře ve formě jednorozměrného vektoru způsobem zdůrazňujícím lokální jasové změny. Hledání takového algoritmu už však přesahuje rámec této práce. Další postup se proto soustředí na jiné metody zpracování obrazu.

6.4 Vyhledávání vad za pomoci hranových operátorů

Jak už bylo zmíněno v části 4, hranové operátory aproximují první nebo druhou derivaci jasové funkce. Tím dojde ke zvýraznění míst, kde se jasová funkce prudce mění. Proto lze hranové operátory rovněž využít k zvýraznění vady v podobě tmavé skvrny.

Použijeme-li jednoduchý hranový operátor, kterým je např. Laplaceův operátor, dojde však také ke zvýraznění prudkých změn jasu, které jsou způsobeny samotnou strukturou roubíku. To ukazuje následující obrázek.



Obrázek 20: Filtrace Laplaceovým operátorem

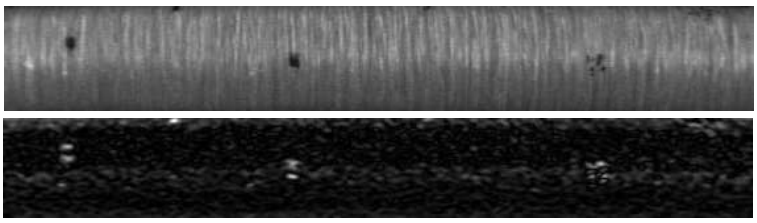
Obrázek roubíku obsahuje množství vertikálních čar a ty se projevují na výsledku hranové detekce. Některé hranové operátory však umožňují dete-

kovat zvlášť hrany v jednotlivých směrech. Jedná se především o Sobelův operátor a operátor Prewittové. Představíme-li si obrázek jako funkci dvou proměnných $z = f(x, y)$, takový operátor je vlastně aproximací parciální derivace funkce podle jedné z proměnných, tedy $\frac{\partial z}{\partial y}$. Pomocí operátoru citlivého pouze v horizontálním směru můžeme potlačit vliv přirozené struktury roubíku, čímž vyniknou ostatní změny jasu, mezi které patří změny způsobené vadami.

Pro zvýraznění vlivu horizontálních změn jasu použijeme filtr, který vychází z operátoru Prewittové. Prvky tohoto filtru ukazuje následující matice:

$$h = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 \end{bmatrix} \tag{25}$$

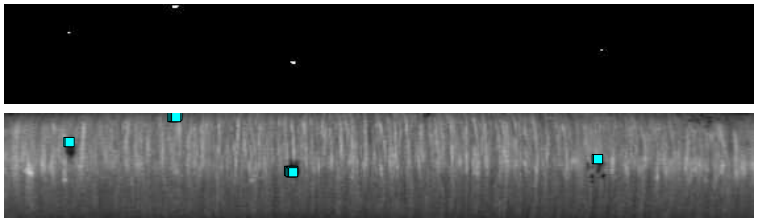
Obrázek 6.4 ukazuje texturu roubíku a výsledek filtrace tímto operátorem. Svislé čáry reprezentující strukturu roubíku jsou potlačeny a vyniknou ostatní lokální extrémy jasu.



Obrázek 21: Filtrace hranovým operátorem

Pro segmentaci skvrn je dále nutné obraz filtrovat za účelem potlačení světlých odlesků. Kulatý tvar roubíku způsobuje odlesky, které je třeba potlačit. Částečnou filtraci zajistí správné použití polarizačního filtru před objektivem kamery už při snímání obrazu. Zbylé světlé body lze odfiltrovat použitím jednoduchého omezujícího filtru. Ten nejprve vypočte typickou hodnotu jasu roubíku (modus) a poté nahradí jasnější pixely touto hodnotou.

Filtrovaný obrázek dále zpracujeme prahováním a převedeme na binární obraz, kde každý pixel (nebo shluk pixelů) s hodnotou „1“. Použitá hodnota prahu tak určuje největší hodnotu, kterou program ještě považuje za přijatelnou pro vyhovující roubík. Výsledek této detekce chyb ukazuje obrázek 6.4.



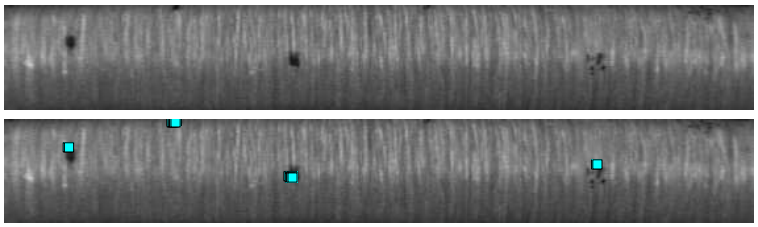
Obrázek 22: Prahování filtrovaného obrázku a zobrazení vad na textuře.

Zhodnocení algoritmu

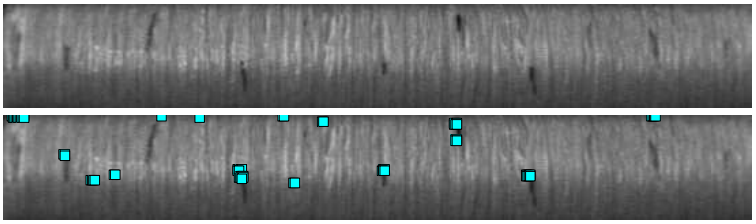
Na prvním z reprezentativních vzorků, který obsahuje velké vady, program správně detekuje všechny nedostatky. U dalších vzorků je však patrná přílišná citlivost algoritmu na tmavé pixely na okrajích obrázku. Upravením rozhodující hodnoty prahu můžeme citlivost algoritmu změnit, je ovšem nutno zvolit rovnováhu tak, aby byl proces stále dostatečně citlivý a zároveň aby nedocházelo k častému chybnému vyřazení vyhovujícího vzorku.

Výhodou tohoto postupu je nízká výpočetní náročnost.

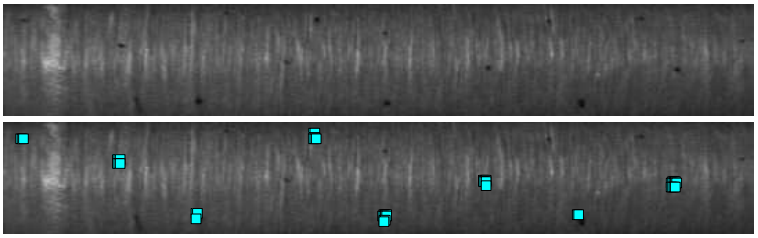
Výsledek hranové detekce na reprezentativních vzorcích



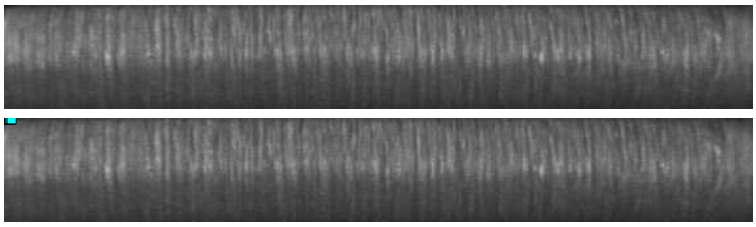
Obrázek 23: Výrazné skvrny.



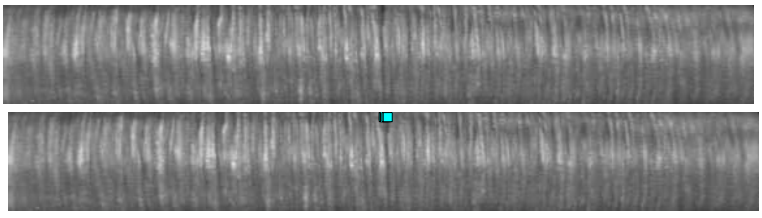
Obrázek 24: Menší skvrny různých velikostí a jasu



Obrázek 25: Drobné skvrny



Obrázek 26: Žádné skvrny, tmavý obraz



Obrázek 27: Žádné skvrny, světlý obraz

6.5 Vyhledávání vad metodami matematické morfologie

Šedotónová morfologie nabízí řadu možností, jak extrahovat důležité informace z obrazu. Morfologický přístup umožňuje najít globální vlastnosti obrazu, které v krajině odpovídají topograficky významným jevům jako jsou údolí, rozvodí, hřebeny hor aj. [1]. Šedotónové transformace kombinují vliv jasových a tvarových prvků obrazu.

Nejprve shrňme problémy, které je třeba při hledání vad vyřešit.

1. Potlačit strukturu roubíku
2. Eliminovat vliv tvaru roubíku
3. Nalézt lokální minima v jednom směru

6.5.1 Potlačení struktury roubíku

K potlačení struktury roubíku lze využít operace šedotónové otevření. Šedotónové otevření umožňuje potlačit jasové špičky v obraze. Jeho výsledek je ve velké míře závislý na použitém strukturním elementu – díky tomu lze jeho vhodnou volbou získat velmi dobrý nástroj pro segmentaci.

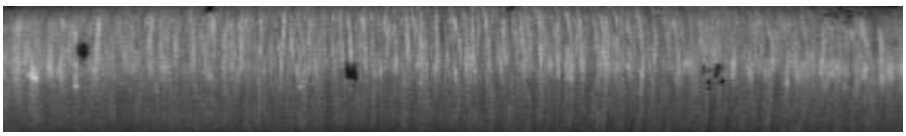
Jako nejvhodnější strukturní element se jeví matice o jednom řádku a mnoha sloupcích, kde všechny hodnoty jsou rovny jedné. Vhodnost použití takového strukturního elementu byla ověřena experimentálně. Obrázek 6.5.1 ukazuje výsledky otevření pro různé podoby strukturního elementu.

Nejlepší výsledek v potlačení jasových špiček můžeme pozorovat na obrázku 6.5.1e), kde bylo použito strukturního elementu 1×30. Zpracování tímto způsobem zapříčiní vyniknutí tmavých nepravidelností povrchu a zároveň začne být dobře patrný jasový přechod způsobený kulatým tvarem objektu. Navíc jsou dobře potlačeny i vertikální tmavé čáry a tím je obrázek dobře připraven pro další zpracování.

6.5.2 Potlačení vlivu tvaru roubíku a nalezení tmavých skvrn

Tam, kde se pomalu mění pozadí, lze použít k segmentaci objektů morfologických operací vrchní část klobouku a spodní část klobouku. Jak je uvedeno v teoretické části práce, tyto operace jsou kombinací množinového rozdílu a otevření či uzavření. Elementárními operacemi jsou proto množinový rozdíl, šedotónová dilatace a šedotónová eroze.

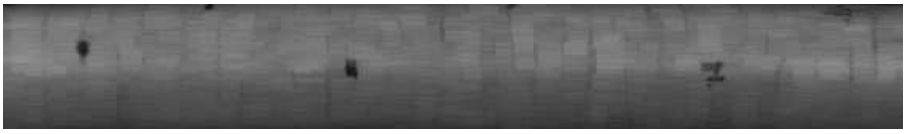
Protože hledané vady jsou tmavé, bylo použito operace spodní část klobouku. Pro ilustraci byla operace rozdělena na uzavření a na množinový



a) originál

b) strukturní element 8×8

c) strukturní element 20×20



d) strukturní element 1×10

e) strukturní element 1×30

f) strukturní element 10×1

g) strukturní element 30×1

Obrázek 28: Ukázka otevření s různými strukturními elementy

rozdíl. Uzavření bylo provedeno stejným strukturním elementem, jako otevření, tedy vodorovnou maticí 1×30 se všemi prvky rovnými jedné. Postup ukazuje obrázek 6.5.2.

Obrázek 6.5.2b, který vznikl uzavřením obrázku 6.5.2a, obsahuje pouze pozadí bez hledaných skvrn. Množinovým rozdílem tohoto obrázku a origi-

a) originál

b) po uzavření strukturním elementem 1×30

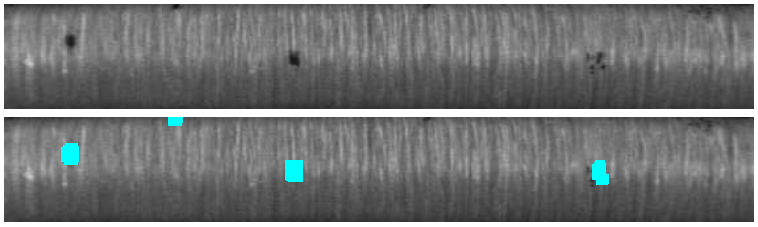
c) množinový rozdíl b) a a)

d) prahování c)

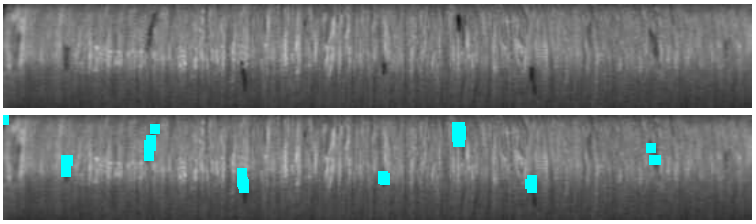
Obrázek 29: Operace spodní část klobouku

nálu tedy získáme právě hledané segmentované vady. Postup doplňuje prahování, které opět zajistí parametrizaci procesu – nastavením prahu zvolíme požadovanou citlivost algoritmu.

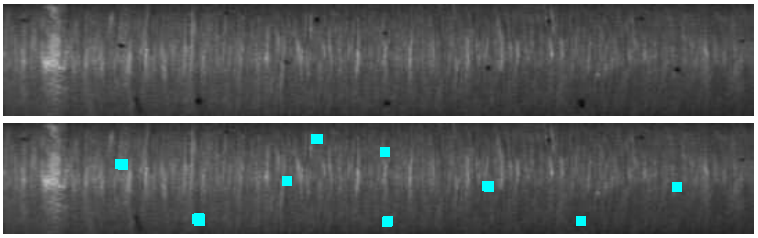
Výsledek morfologické detekce na reprezentativních vzorcích



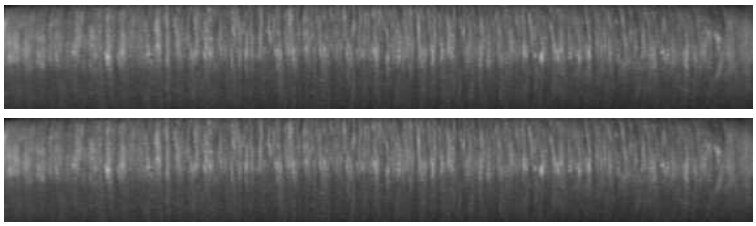
Obrázek 30: Výrazné skvrny.



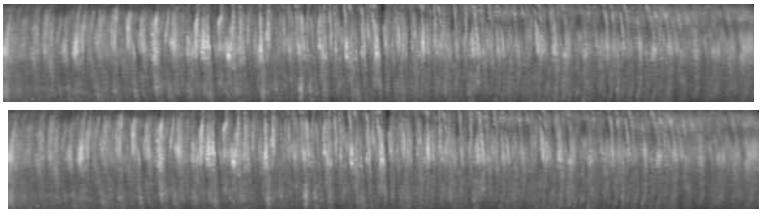
Obrázek 31: Menší skvrny různých velikostí a jasu



Obrázek 32: Drobné skvrny



Obrázek 33: Žádné skvrny, tmavý obraz



Obrázek 34: Žádné skvrny, světlý obraz

6.5.3 Testování algoritmu na vzorcích

Pro testování algoritmu vyhledávání vad byly použity vzorky roubíků dodané od výrobce. Jednalo se o roubíky menších rozměrů typu SEUE a větších rozměrů, typ MEUT[1].

Pro snímání vzorků byla použita kamera s televizní normou a výstupním rozlišením 768×576. Použitý objektiv měl nastavitelnou ohniskovou vzdálenost 6-15 mm a byl nastaven na maximální zvětšení (15 mm). Dále byl opatřen polarizačním filtrem pro potlačení odlesků roubíku.

Počet testovaných vzorků: 25 bezchybných, 45 s vadami.

vzorek	počet	zn. vyhovující	ozn. vadné	% úspěšnosti
vyhovující	25	24	1	96 %
vadný	45	3	43	93,3 %
celkem	70	27	43	94,3 %

Tabulka 1: Test hledání vad pomocí šedotónové morfologie

Při testování bylo zjištěno, že algoritmus je velmi citlivý na jasově výrazné vady i těch nejmenších rozměrů (2×2 pixely). Přesvědčit se o tom můžeme na obrázku 31. Vzhledem k tvarům použitých strukturních elementů má však algoritmus problémy detekovat vady v podobě dlouhých horizontálních šmouh.

Výsledné statistiky mohou být dále ovlivněny změnou prahu. Nastavíme-li práh na menší hodnotu, zvyšujeme tím citlivost algoritmu na chyby. Ovšem samozřejmě je to za cenu většího počtu správných výrobků, které jsou identifikovány jako vadné.

parametr	hodnota
strukturní element otevření	1×30
strukturní element uzavření	1×50
práh	33

Tabulka 2: Parametry algoritmu pro test

[1] Vzhledem k tomu, že nebyl k dispozici dostatek vzorků, na kterých by se vyskytovaly výrobní vady, byla část vzorků uměle znečištěna barvivem.

6.6 Výpočetní náročnost morfologických operací

Elementárními operacemi pro výše zmíněný postup jsou šedotónová eroze a šedotónová dilatace. Pro otevření je nutno provést erozi a dilataci, pro spodní část klobouku dilataci, erozi a množinový rozdíl. Vzhledem k tomu, že je nutno pro každý roubík zpracovat obrázek minimálně třikrát, jedná se o dvanáct provedení eroze nebo dilatace pro jeden roubík. Snahou je proto tyto operace provádět co nejefektivněji.

Podle definice lze vypočíst erozi hledáním minima a dilataci hledáním minima ve vektoru o délce, která je rovna počtu prvků s hodnotou 1 v použitém strukturním elementu. V případě eroze je tedy nutno pro každý pixel výstupního obrazu vypočítat minimum z vektoru hodnot. Náročnost operace se tak zvyšuje s velikostí použitého strukturního elementu.

Při použití strukturního elementu v podobě jednoho řádku však lze výpočet minima vektoru značně zjednodušit. Situaci si lze představit jako následující obrázek:

... 14 12 [**14 13 15 16 15 13 15**] 12

Obrázek 35: Posuvné okno pro výpočet minima

...

Uvnitř okna odpovídajícího strukturnímu elementu je vypočteno minimum a okno se posouvá o jeden prvek doprava. Změní se tedy pouze dva prvky pole, ze kterého je minimum počítáno. Proto můžeme použít algoritmy umožňující za těchto podmínek efektivní výpočet.

6.6.1 Třídění haldou

Třídění haldou je výhodné právě tehdy, potřebujeme-li rychle získat maximum nebo minimum pole. V struktuře haldy je minimum (respektive maximum) vždy na jejím začátku. Rychlost setřídění haldy je úměrná $O(n \log n)$ Po jejím sestavení je výpočetní náročnost přidání nebo ubrání prvku úměrná logaritmu počtu prvků $O(\log n)$. Při délce SE 50 prvků by na každý pixel bylo potřeba přibližně 6 porovnání. Podrobný popis rychlé verze algoritmu třídění haldou lze najít v [7].

6.6.2 van Herkův algoritmus

van Herkův algoritmus [6] umožňuje vypočítat pole minimálních (respektive maximálních) hodnot, které vyžadují šedotónové morfologické operace, bez vlivu velikosti strukturního elementu na výpočetní náročnost. Lze ho použít pro řádkové nebo obdélníkové strukturní elementy. Výpočetní náročnost,

kterou uvádí prameny ([6]) je ještě nižší než v případě použití haldy – na jeden výstupní pixel je v nejhorším případě třeba tří porovnání hodnot.

Použití algoritmu třídění haldou a van Herkova algoritmu nebylo v rámci práce testováno – postup byl realizován s použitím funkcí pro šedotónovou morfologii programu MatLab. Zdrojový kód v jazyce MatLab lze najít v příloze diplomové práce.

Závěr

V rámci diplomové práce bylo navrženo a vyzkoušeno několik postupů detekce nečistot stravitelných střev metodami analýzy obrazu. Důraz byl kladen na spolehlivé odhalení vad, které se mohou náhodně vyskytnout během výroby, robustnost řešení a realizovatelnost detekce v krátké době tak, aby se mohla stát součástí kontrolního systému, který bude pracovat v reálném čase.

Tři různé algoritmy byly realizovány s pomocí jazyka C++ a programu MatLab a vyzkoušeny na reprezentativních vzorcích výrobků. Detekce nečistot na základě operací matematické morfologie se ukázala jako nejspolehlivější a zároveň dostatečně robustní. Byla proto dále podrobena rozsáhlejšímu testování na sedmdesáti vzorcích, které vykazovaly různé množství vad z výroby i umělých, a které se skládaly z několika druhů výrobků různé velikosti a struktury povrchu. Obrázky vzorků byly pořízeny v laboratoři zpracování obrazu s použitím kamery s televizní normou, různých osvětlovačů a optických filtrů.

V průběhu laboratorního testování bylo označeno 96 % výrobků, které nevykazovaly žádnou vadu správně a zbylá 4 % byla označena jako vadná. Z výrobků obsahujících alespoň jednu nečistotu bylo 93,3 % správně označeno jako nevyhovující a zbylých 6,7 % nebylo odhaleno a tyto výrobky prošly kontrolou jako vyhovující. Dosažené hodnoty odpovídají obvyklým výsledkům typickým pro analýzu obrazu.

Na základě práce lze doporučit sestavení systému kontroly kvality, který bude založen na snímání roubíků kamerou na výstupu řásnění a zpracování obrazu metodami šedotónové morfologie.

Literatura

[1] Hlaváč V., Sedláček M.: *Zpracování signálů a obrazů*. Vydavatelství ČVUT, 2005. 255 s. ISBN 80–01–03110–1

[2] Fischer J.: *Optoelektronické senzory a videometrie*. Vydavatelství ČVUT, 2002. 143 s. ISBN 80–01–02525–X

[3] Hlaváč V., Šonka M.:*Počítačové vidění*. Grada, 1992. 272 s. ISBN 80–85424–67–3

[4] Schatzmann J.:*Using Self-Organizing Maps to Visualize Clusters and Trends in Multidimensional Datasets* [online]. 2003. Dostupný z WWW: <http://mi.eng.cam.ac.uk/ js532/papers/schatzmann03soms.pdf>

[5] Taner M.:*Kohonen's self-organizing networks with "conscience"* [online]. 1997. [cit. 2006–02–21]. Dostupný z WWW: <http://www.rocksolidimages.com/pdf/kohonen.pdf>

[6] van Herk M.:*A fast algorithm for local minimum and maximum filters on rectangular and octagonal kernels*. Pattern Recognition Letters, 1992. 517–521 s. ISSN 0167–8655

[7] Edelkamp S., Wegener I. :*On the performance of WEAK–HEAPSORT*. ECCC, 1999. 16 s. ISSN 1433–8092

[8] Mäenpää T., Turtinen M., Pietikäinen M. :*Real-Time Surface Inspection by Texture*. [online]. 2003. [cit. 2006–02–21]. Dostupný z WWW: <http://www.ee.oulu.fi/mvg/publications/>

Seznam obrázků

Příloha A - zdrojové kódy v MatLabu

```
% ------------------------------------------------
% najdi_prah.m
%
% Programek pro segmentaci objektu roubiku z obrazku.
% Zvoli prah na zaklade extremu v histogramu jasovych hodnot.
% ------------------------------------------------

clear all
close all

% otevri soubor
[filename, pathname] = uigetfile('*.png','Otevři obrázek');

% nacti obrazek
A = imread([pathname filename]);

%zobraz
figure(1);
imshow(A);

% histogram
figure(2)
[counts,x] = imhist(A);
imhist(A);

% najdi lokalni minimum: filtrace a derivace histogramu
fB = fir1(30, 0.1);
fcounts = filtfilt(fB,1,counts);
dcounts = diff(fcounts);
len = length(dcounts);
for j = 50:200,
   if (dcounts(j)<=0)&(dcounts(j-1)>=0),
      break
   end
end

disp('prah: ');
j
```

```
% prahovani
bwA = im2bw(A, (j-10)/255);
figure(3)
subplot(2,1,1);
imshow(bwA)

% otevreni a vyber
filteredA = erode(bwA,ones(50,50));
[i,j] = find(filteredA);
filteredA = bwselect(bwmorph(bwA,'open'), j(1),i(1), 4);
filteredA = bwmorph(filteredA,'close');
figure(3);
subplot(2,1,2);
imshow(filteredA)


% ------------------------------------------------
% detekce.m
%
% Programek pro detekci necistot na roubiku pomoci sedotonove
% morfologie. Vstupem je oriznuty obrazek roubiku.
% ------------------------------------------------

clear all
close all

% otevri soubor
[filename, pathname] = uigetfile('*.bmp','Otevři obrázek');
A = imread([pathname filename]);
%orez kvuli ruzkum
[x,y,c] = size(A);
A = A(3:x,5:y-5,1);
%strukturni element
SE = ones(1,30);
%otevreni
B = imopen(A,SE);
%bottom hat
C = imbothat(B,ones(1,50));
```

```
%prahovani
D = im2bw(C,0.13);

subplot(5,1,1)
imshow(A);
subplot(5,1,2)
imshow(B);
subplot(5,1,3)
imshow(C);
subplot(5,1,4)
imshow(D);
subplot(5,1,5)
imshow(A);

%zobraz identifikované chyby
patchSize = 7;
patchX = [1,1,-1,-1]*patchSize/2;
patchY = [-1,1,1,-1]*patchSize/2;
[sizeX,sizeY]=size(D);
for i=1:sizeX,
    for j=1:sizeY,
        if( D(i,j) ) > 0,
            h = patch(patchX+j,patchY+i,'c');
            set(h,'LineStyle','none');
        end
    end
end
```